Пояснювальна записка
до магістерської дисертації
на тему: «Система управління ресурсами високонавантажених сервісів в IT-інфраструктурі»

Київ – 2018 рік

РЕФЕРАТ

Магістерська дисертація: 90 с., 27 рис., 28 табл., 23 джерел.

Актуальність теми. Зараз більшість населення Землі завдяки розвитку інформаційних технологій (ІТ) та телекомунікаційних мереж, а також повсюдному доступу до інтернет мають можливість отримувати ІТ-послуги в будь-якому місті. ІТ -оператори змушені постійно підвищувати якість наданих ІТ-послуг, оскільки користувачі зацікавлені не лише в отриманні ІТ-послуг, їх також цікавлять якість, асортимент та обсяг наданих послуг. Для того, щоб залишитися на ринку компаніям які надають послуги ІТ-інфраструктури потрібно постійно покращувати обладнання та програмне забезпечення, яке використовується, без цього їх з легкістю обійдуть конкуренти. Для операторів постійним актуальним завданням є підвищення якості ІТ-послуг, або підтримання якості на узгодженому рівні незалежно від динаміки росту кількості користувачів окремого сервісу. Оператори ІТ-послуг покращують обслуговування користувачів шляхом підвищення швидкості телекомунікаційної мережі, балансуванням навантаження та вирішенням проблем з продуктивністю.

Ефективність функціонування інформаційних систем та вартість ІТ-послуг суттєво залежать від потужності ІТ-інфраструктури та ефективності використання її ресурсів. Основу ІТ-інфраструктури центрів оброблення даних (ЦОД) складають фізичні сервери, на які встановлюють віртуальні машини (ВМ). ІТ-послуги можуть надаватися за допомогою одного або декількох ЦОД.

Для надання користувачам ІТ-послуг на ВМ, які використовують обчислювальні ресурси ІТ-інфраструктури ЦОД, встановлюють різноманітні застосування. Збільшення запитів користувачів призводить до погіршення якості надання ІТ-послуг. У цьому випадку для підтримання якості послуг на узгодженому рівні система управління ІТ-інфраструктурою (СУІ), використовуючи методи масштабування ресурсів, їх розподілу та перерозподілу додає відповідним додаткам або ВМ, що підтримують високонавантажені сервіси, додаткові об’єми обчислювальних ресурсів.

Актуальність завдання підвищення ефективності роботи СУІ обумовлена високою вартістю обчислювальних, комунікаційних ресурсів та ресурсів для збереження

даних у ЦОД, а також суттєвою залежністю доходів операторів ІТ-послуг від задоволеності користувачів якістю наданих послуг. Тому дана робота присвячена розробленню методів управління якістю послуг високонавантажених ІТ-інфраструктур, розподілом навантаження та збільшення потужності додатків шляхом масштабування обчислювальних ресурсів.

Метою роботи є розроблення методів для управління ресурсами та керування якістю надання послуг для користувачів високонавантажених сервісах в ІТ-інфраструктурі з підвищенням ефективності використання ресурсів в центрах обробки даних.

Об'єктом дослідження є система масштабування ресурсів, їх розподіл та перерозподіл з додаванням додаткових ресурсів для підтримання якості наданих ІТ-послуг.

Предметом дослідження є масштабування та розподіл ресурсів, що допоможе забезпечити якість надання послуг.

Методи дослідження включають в себе моделювання ПЗ, дослідження, обчислення, теорія ймовірності та обчислювальні технології.

Практичне значення результатів.

Розроблено математичні моделі та методи масштабування ресурсів, як основа управління ресурсами високонавантажених сервісів в ІТ-інфраструктурі. На основі одержаних в роботі результатів було розроблено рішення для розподілу ресурсів та керування якістю надання послуг для користувачів. Робота може бути використана в СУ.

ВИСОКОНАВАНТАЖЕНІ СИСТЕМИ, ІТ-ІНФРАСТРУКТУРА, УПРАВЛІННЯ РЕСУРСАМИ, ЦЕНТРИ ОБРОБКИ ДАНИХ, СЕРВЕРА

ABSTRACT

Master's thesis: 84 pages, 27 figures, 28 tables, 23 sources.

Actuality of theme. Today, most of the Earth's population, thanks to the development of information technology (IT) and telecommunication networks, as well as widespread Internet access, have the opportunity to receive IT services in any city. IT service providers are forced to constantly improve the quality of IT services, as users are interested not only in obtaining IT services, but also want to receive high – quality IT services, the assortment and volume of which is increasing every day. Therefore, for the operators, the task of improving the quality of IT services, or maintaining quality at an agreed level, regardless of the dynamics of the number of users of a separate service, is constantly relevant. IT service operators improve user services by increasing the speed of the telecommunication network, balancing the load and solving productivity problems.

The effectiveness of the functioning of information systems and the cost of IT services are significantly dependent on the capacity of the IT infrastructure and the efficiency of its use of resources. The new IT data center infrastructure (data centers) is made up of physical servers, which are installed by virtual machines (VMs). IT services may be provided by one or more data centers.

To provide IT users with IT services that use computing resources of IT infrastructure, a variety of applications are available for VMs. Increasing user queries leads to a deterioration in the quality of IT services. In this case, to maintain the quality of services at an agreed level, the IT system management system, using methods of resource scaling, distribution and redistribution, adds additional computational resources to relevant applications or VMs that support high – volume services.

The urgency of the task of increasing the efficiency of the VSI is due to the high cost of computing, communication resources and resources for data storage in the data centers, as well as the significant dependence of the revenues of IT service providers on the satisfaction of users of these services with their quality. Therefore, this work is devoted to the development of a method for managing the quality of services of high – load IT infrastructures

by distributing the load and increasing the power of applications by scaling computing resources.

The goal of the work is to develop methods for managing resources and managing the quality of service provision for users of high-capacity services in the IT infrastructure, with increased efficiency in the use of resources in data centers. The object of the research is the system of scaling resources, their distribution and redistribution to add additional resources to maintain the quality of providing IT services.

The subject of the study is the scaling and distribution of resources, which will help ensure the quality of service delivery.

Research methods include software simulation, research, computing, probability theory, and computing technologies.

The practical value of the results developed in this paper, mathematical models and methods of scaling resources, as the basis for the distribution of virtual machines in data centers.

HIGH-AVAILABLE SYSTEMS, IT-INFRASTRUCTURE, RESOURCE MANAGEMENT, DATA PROCESSING CENTERS, SERVER

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АЗ – апаратні засоби

ВМ – віртуальна машина

ГА – генетичний алгоритм

ІТ – інформаційні технології

ОП – оперативна пам'ять

ПЗ – програмне забезпечення

СУІ – система управління інфраструктурою

СОА – сервіс-орієнтована архітектура

ФС – фізичний сервер

ЦОД – центр обробки даних

API (Application Programming Interface) – інтерфейс прикладного програмування

DDD (Domain-driven design) – набір принципів спрямованих на створення оптимальних додатків

CRM (Customer Relationship Management) – управління взаємовідносинами з клієнтами

FTP (File Transfer Protocol) – протокол для передачі файлів

ВСТУП

Зараз більшість населення Землі завдяки розвитку інформаційних технологій (ІТ) та телекомунікаційних мереж, а також повсюдному доступу до інтернет мають можливість отримувати ІТ-послуги в будь-якому місті. ІТ -оператори змушені постійно підвищувати якість наданих ІТ-послуг, оскільки користувачі зацікавлені не лише в отриманні ІТ-послуг, їх також цікавлять якість, асортимент та обсяг наданих послуг. Для того, щоб залишитися на ринку компаніям які надають послуги ІТ-інфраструктури потрібно постійно покращувати обладнання та програмне забезпечення, яке використовується, без цього їх з легкістю обійдуть конкуренти. Для операторів постійним актуальним завданням є підвищення якості ІТ-послуг, або підтримання якості на узгодженому рівні незалежно від динаміки росту кількості користувачів окремого сервісу. Оператори ІТ-послуг покращують обслуговування користувачів шляхом підвищення швидкості телекомунікаційної мережі, балансуванням навантаження та вирішенням проблем з продуктивністю.

Ефективність функціонування інформаційних систем та вартість ІТ-послуг суттєво залежать від потужності ІТ-інфраструктури та ефективності використання її ресурсів. Основу ІТ-інфраструктури центрів оброблення даних (ЦОД) складають фізичні сервери, на які встановлюють віртуальні машини (ВМ). ІТ-послуги можуть надаватися за допомогою одного або декількох ЦОД.

Для надання користувачам ІТ-послуг на ВМ, які використовують обчислювальні ресурси ІТ-інфраструктури ЦОД, встановлюють різноманітні застосування. Збільшення запитів користувачів призводить до погіршення якості надання ІТ-послуг. У цьому випадку для підтримання якості послуг на узгодженому рівні система управління ІТ-інфраструктурою (СУІ), використовуючи методи масштабування ресурсів, їх розподілу та перерозподілу додає відповідним додаткам або ВМ, що підтримують високонавантажені сервіси, додаткові об’єми обчислювальних ресурсів.

Актуальність завдання підвищення ефективності роботи СУІ обумовлена високою вартістю обчислювальних, комунікаційних ресурсів та ресурсів для збереження даних у ЦОД, а також суттєвою залежністю доходів операторів ІТ-послуг від задоволеності користувачів якістю наданих послуг. Тому дана робота присвячена розробленню методів управління якістю послуг високонавантажених ІТ-інфраструктур, розподілом навантаження та збільшення потужності додатків шляхом масштабування обчислювальних ресурсів.

Метою роботи є розроблення методів для управління ресурсами та керування якістю надання послуг для користувачів високонавантажених сервісах в ІТ-інфраструктурі з підвищенням ефективності використання ресурсів в центрах обробки даних.

Об'єктом дослідження є система масштабування ресурсів, їх розподіл та перерозподіл з додаванням додаткових ресурсів для підтримання якості наданих ІТ-послуг.

Предметом дослідження є масштабування та розподіл ресурсів, що допоможе забезпечити якість надання послуг.

Методи дослідження включають в себе моделювання ПЗ, дослідження, обчислення, теорія ймовірності та обчислювальні технології.

Практичне значення результатів.

Розроблено математичні моделі та методи масштабування ресурсів, як основа управління ресурсами високонавантажених сервісів в ІТ-інфраструктурі. На основі одержаних в роботі результатів було розроблено рішення для розподілу ресурсів та керування якістю надання послуг для користувачів. Робота може бути використана в СУ.

1 ХАРАКТЕРИСТИКА ОБ'ЄКТУ УПРАВЛІННЯ

Необхідність підвищення ефективності виконання бізнес-процесів та процесів діяльності вимагає розвитку інформаційних технологій. Сьогодні все активніше використовується парадигма хмарних обчислень, яка має на увазі централізоване оброблення інформації в ЦОД. ЦОД забезпечує надання ІТ-послуг для автоматизації бізнес-процесів. Для підтримання високої якості послуг та зменшення витрат на адміністрування в ЦОД використовуються СУІ. СУІ здійснюють масштабування високонавантажених інформаційних систем, запускають додатки, виконують балансування навантаження, провадять моніторинг компонентів ІТ-інфраструктури таких як сервери, пам'ять, процесори, здійснюють контроль операцій введення та виведення й можливість доступності сервісу. Одним із основних завдань СУІ є контроль функціонування ІТ-сервісів та якості, з якою вони надаються користувачам.

Інформаційні системи для яких важлива швидкість виконання запитів до сервісів, називаються високопродуктивними системами, а системи (сервіси) з великою кількістю користувачів – високонавантаженими системами (сервісами). Перевантаження таких систем починається тоді, коли сервер не може здійснювати обробку даних протягом певного періоду часу. В системах з послідовною обробкою вхідних запитів кожен наступний запит очікує завершення обробки всіх попередніх запитів. Таким чином, якість надання послуг для користувача погіршується.

Основним методом забезпечення високої якості високонавантажених сервісів з боку СУІ є масштабування ресурсів, які виділені для застосувань, що надають цей сервіс. Масштабування забезпечує здатність системи, мережі або процесу обробляти зростаючу кількість запитів без погіршення якості послуг.

На серверах ЦОД розташовуються ВМ. Для кожної ВМ виділяються певні об'єми ресурсів сервера. На ВМ встановлюється один або декілька додатків відповідно до виділених ресурсів. При роботі додатків виникають операції введення-виведення, до яких відносять будь-яку передачу даних в ЦП або з нього.

Так зчитування даних з дисководу також розглядається як операції введення- виведення [1]. Серед набору схем, що підтримують ЦП є схеми для здійснення операцій введення/виведення при взаємодії з пам'яттю та іншими пристроями.

Швидкість виконання операцій введення/виведення, швидкодія процесору та об’єм виділеної оперативної пам’яті, безпосередньо впливають на продуктивність додатків. Ці показники найчастіше використовують в СУІ для оцінки якості ІТ-сервісів за непрямими ознаками. Тому саме показники, що вказують на завантаженість ресурсів обрані для управління якістю сервісів високонавантажених ІТ-інфраструктур.

1.1 Особливості використання ЦОД

Центр оброки даних представляє собою будівлю, де знаходиться серверне та мережеве обладнання. Основне завдання таких центрів – обробка великої кількості інформації, можливість збереження даних та швидкість виконання запитів до сервісів.

Популярність використання центрів обробки даних зростає з кожним днем. Центри обробки даних дозволяють вирішити такі проблеми:

– зростає потік інформації, який сервери повинні обробляти;

– зростає кількість користувачів, що використовують систему;

– споживачі використовують одні й ті самі дані в різних куточках землі.

Тенденція використання центрів обробки даних показує, що їх використовують лише великі компанії, що мають потужні інформаційні системи планування ресурсів. В таких компаніях об’єм інформації великий наскільки, що його не реально обробити використовуючи власні ресурси компанії.

Центри обробки даних будуть корисні при вирішенні будь-яких проблем з ІТ-інфраструктурою. Послуги, які можуть надаватись ЦОД, крім відмовостійкості:

– оренда стійок;

– розміщення серверів;

– підключення інтернету;
– оренда каналів зв’язку;
– встановлення ПЗ;
– налаштування ПЗ;
– адміністрування.

Додаткові послуги, що можуть надаватись центрами обробки даних для компаній:

– оренда потужності;
– дисковий простір за допомогою якого можна роботи резервне копіювання в системі;
– технічне обслуговування.

ЦОД гарантують збереження інформації. Інформація являє собою, досить великий та дорогоцінний ресурс. При використанні центрів обробки даних можемо отримати наступні переваги:

– економія коштів на створення та впровадження проектів;
– економія на купівлі серверів, стійок та оренді приміщень;
– можливість масштабування ресурсів протягом короткого часу;
– безпечність зберігання ресурсів;
– управління може відбуватись в декількох країнах чи містах;

1.2 Типи ЦОД

Перед тим, як обрати вид центра-обробки даних, який потрібен для вирішення проблеми, потрібно чітко розуміти для чого потрібні дата-центри та як вони можуть бути використанні для вирішення проблем в IT-інфраструктурі. В залежності від обраного виду центру обробки даних, його можливості та ціна може змінюватися.

Основними видами центрів оброки даних є: комерційні, провайдери керованих послуг, колокаційні, масові або оптові дата- центри, виділені хостинги та керовані хостинги.

Існує багато видів хостигу і кожен з них підходить для вирішення певної задачі. Розглянемо види центрів-обробки даних детальніше:

– внутрішні центри обробки даних представляють собою великі організації, які розробляють та створюють свої власні об’єкти хостингу. Потужність обладнання буде залежати від інвестицій, які фірма зробить для досягнення масштабування;

– корпоративні центри обробки даних виконують обробку запитів та слугують для вирішення проблем в бізнесі;

– колокаційні дата-центри надають можливість орендувати частину дата-центру декільком компаніям. Основна ціль – контроль апаратних засобів і можливість надаванням стороннім організаціям, і внутрішнім системам можливість оренди серверів в центрах обробки даних;

– оптові дата-центри представляють собою провайдери, що забезпечують клієнтів ресурсами, проте оптові дата-центри пропонують можливість обробки більших об’ємів інформації на відміну від колокаційних датацентрів;

– виділений хостинг, де клієнт має можливість орендувати сервери. Система може підтримувати якість наданих послуг незалежно від навантаження, яке створюють користувачі системи, та вартість такого хостингу значно вища;

– керований хостинг представляє собою систему, в якій головний елемент – провайдер. Провайдер використовується для керування серверами та може бути використаний для адміністративних послуг. Апаратне забезпечення, що надається в оренду може бути використане для будь-яких цілей.

1.3 Типова архітектура ЦОД

Центр оброки даних представляє собою будівлю для обчислення та зберігання даних. Інфраструктура обслуговує потреби користувачів центрів обробки даних та здійснює керування ресурсами при навантаженнях. Центри обробки даних можуть здійснювати балансування навантаження та вирішення проблем з продуктивністю. Для забезпечення діяльності ЦОД необхідна складна та масштабована архітектура. Приклад архітектури ЦОД описаний в роботі [1].

Багаторівнева архітектура корисна в високонавантажених та великих системах. Підхід з багаторівневою архітектурою використовується для створення гнучких додатків шляхом розбиття додатка на рівні. Це дозволяє розробникам модифікувати або додати певний рівень, не змінюючи всю архітектуру додатку.

Основні рівні, які представлені в архітектурі центра обробки даних: головний рівень, рівень агрегації та рівень доступу. Рівні архітектури описані в роботі [2].

Головний рівень слугує для взаємодії з іншими модулями за допомогою рівня агрегації. Для комунікації з іншими рівнями використовуються протоколи: OSPF або EIGRP. Також, в центрах обробки даних повинна бути підсистема балансування навантаження, яка використовує алгоритми розподілення ресурсів.

Для взаємодії з мережею використовуються рівень доступу, що представляє собою серверні компоненти, які поділяються на комутатори та на контролюючі сервери.

Багаторівневий підхід – головна основа дизайну центру обробки даних, мета якого – поліпшити масштабованість, продуктивність і обслуговування.

1.4 Принцип роботи ЦОД

Центри обробки даних використовують віртуалізацію, кластеризацію, масштабування та резервування.

Віртуалізація є основною технологією яка використовується для побудови центрів обробки даних. За допомогою віртуалізації з’являється можливість масштабувати ресурси [3] та зберігати час для запуску операційної системи з уже встановленим програмних забезпеченням. У віртуалізації є чітко розділені рівні, які слугують для ізоляції обчислювальних ресурсів, додатків та мережі. Рівні віртуалізації зображені у додатку В.

Віртуальна машина представляє собою файл даних, який виконується та може бути переміщений на будь-який комп’ютер і на різних комп’ютерах буде мати однакову поведінку.

Ресурси розподіляються по мірі необхідності від фізичного до віртуального середовища. Користувачі системи не помічають, що вони працюють у віртуалізованому середовищі. Основою віртуалізації є сервери, які використовують програмний рівень (так званий гіпервізор), щоб емулювати фізичне обладнання. Сюди може входити пам’ять, процесор, пристрої введення та виведення.

При використані фізичних серверів, операційна система взаємодіє зі справжнім обладнанням. Використовуючи віртуалізацію, операційна система взаємодіє вже не зі справжнім, а з емульованим обладнанням. Таким чином, віртуалізовані операційні системи не знають, що вони працюють на віртуалізованому обладнанні в центрах обробки даних.

Віртуалізація працює в центрах обробки даних тому, що більшість операційних систем та програм не потребують повного використання характеристик основного апаратного забезпечення [5]. Це надає можливість мати більшу гнучкість, контроль та ізоляцію, вилучаючи залежність від даної апаратної платформи.

1.5 Віртуалізація в системах управління

Кожна з систем, що використовується для управління системою повинна мати набір модулів [6], які будуть використані для вирішення завдань в системі управління. Система для розподілу навантажень, повинна включати в себе такі модулі: модуль управління та модуль аналізу, диспетчер запитів та додатки для здійснення моніторингу в системах. Призначення модулів системи управління:

– диспетчер запитів – модуль, який знаходиться в системі управління і пропускає через себе запити від користувача. Чітко контролює дії, що приходять від користувачів та має можливість спілкуватись з модулем моніторингу для того, щоб скласти історію запитів або зробити аналіз системи;

– модуль моніторингу представляє собою частину системи, яка слугує для створення метрик. Також може відслідковувати навантаження в системах та на агентах. Взаємодіє з модулем аналізу, щоб правильно проаналізувати інформацію, яка отримана з модуля моніторингу;

– модуль аналізу слугує для аналізу даних та метрик, що були отримані з модуля моніторингу. Може роботи, додаткові аналізи запитів, які будуть впливати на розподілення ресурсів в подальшому;

– модуль управління допомагає генерувати завдання для системи управління, базуючись на модулі аналізу та модулі моніторингу, відповідає за розподілення ресурсів;

– агенти використовуються в системі управління для виконання команд на віртуальних машинах. Також, агенти збирають інформацію про завантаженість віртуальних машин в центрах обробки даних.

Система розподілу ресурсів може бути реалізована, у вигляді програмного забезпечення, яке встановлюється в систему управління. Сервери, що знаходяться в системі управління допомагають керувати об’єктами на апаратному та програмному рівні. Кожен сервер, що знаходиться в центрі обробки даних складається з віртуальних машин та гіпервізора.

Одиницею центрів обробки даних є сервер, на кожному сервері встановлені віртуальні машини. При встановлені додаткових віртуальних машин на серверах, сума фізичних характеристик всіх віртуальних машин не повинна перевищувати характеристики сервера.

Гіпервізор слугує для обслуговування декількох операційних систем на одному сервері. Можливість виконання операцій з різними операційними системами на серверах описано в роботі [7]. Гіпервізор, також, може виконувати функцію ізоляції операційної системи в центрі обробки даних.

На даний момент, всі сервери мають можливість запустити лише одну операційну систему. Для того, щоб запускати декілька різних операційних систем на одному сервері була створена віртуалізація. Використовуючи віртуалізацію, на сервер можна встановлювати декілька віртуальних машин з різними операційними системами.

Центри обробки даних слугують для оброблення великої кількості інформації та великих навантажень. Для будь-якої інфраструктури важливо обслуговувати будь-які навантаження в інфраструктурі. Тому потрібно чітко розуміти,

що можна зробити для надання якісного обслуговування користувачам. Завантаженість в центрах обробки даних може відбуватись на різних серверах в різний час. Для того, щоб контролювати завантаженість системи, в центрах обробки даних повинна бути система управління, яка буде розподіляти та контролювати ресурсів на серверах.

Як правило в мережах є адміністратор, який слідкує за розподілом навантаження у системі. Управління ресурсами описано в роботах [16-17]. Якщо в системі забагато ресурсів, то можна здійснити операцію перерозподілу ресурсів для віртуальних машин чи серверів.

При додаванні нового сервера в центр обробки даних, сервер немає віртуальних машин. Тільки за необхідності додаткових ресурсів або при балансуванні навантаження на серверах буде створена віртуальна машина. Модуль управління може створити додаткові віртуальні машини на які буде переправлена частина навантаження.

Віртуальні машини мають ряд переваг над фізичними серверами. Однією з них є ізоляція. Ізоляція надає можливість створення окремого середовища, яке складається з операційної системи і додатка.

Головна проблема дата-центрів – неоптимальне використання наявних обчислювальних потужностей. Часто трапляються ситуації, коли ресурсів не вистачає, або їх забагато. Віртуалізація допомагає в агрегуванні всіх віртуальних машин на серверах і дає можливість запускати декілька додатків на різних операційних системах на одному сервері.

1.6 Система управління як частина ЦОД

Центри обробки даних включають в себе велику кількість серверів, де кожний сервер може керувати своїми віртуальними машинами для надання послуг користувачам. Для керування ресурсами в центрі обробки даних використовується модуль агрегації, який розподіляє навантаження між серверами для на-

дання якісного обслуговування користувачам. Управління корпоративною IT-інфраструктурою описано в роботах [18-19]. Схема для управління навантаженням в ЦОД зображена на рис.1.2.

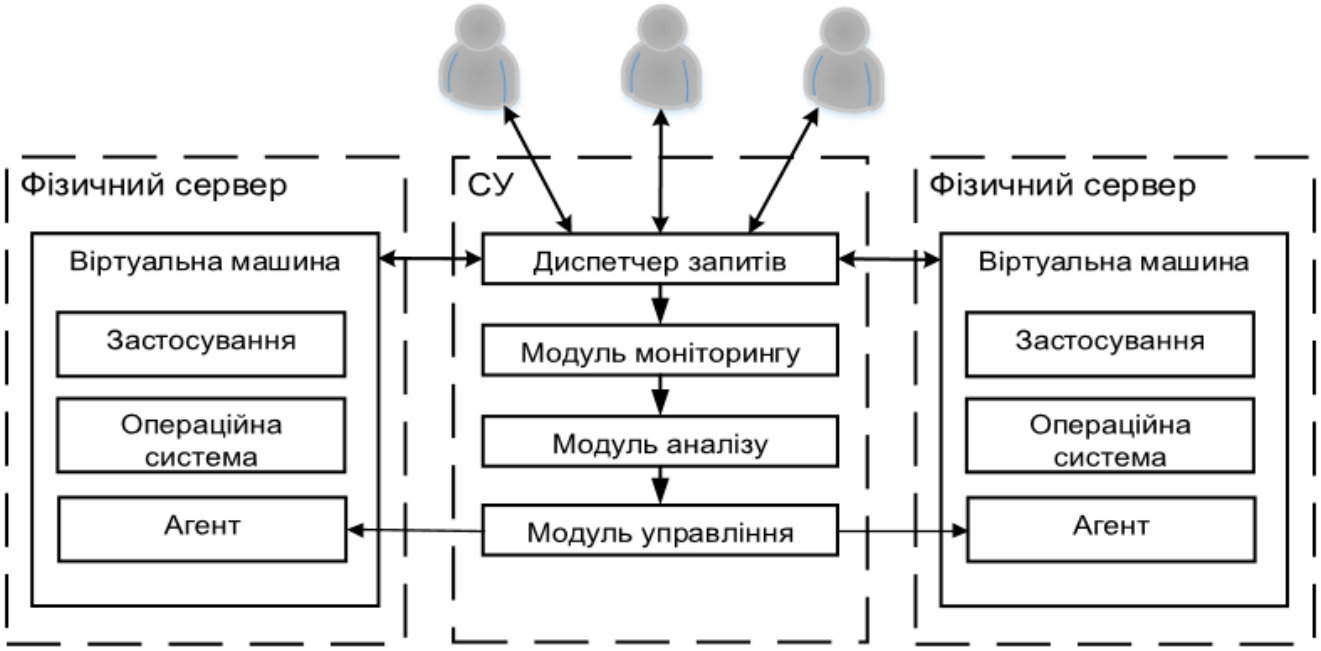


Рисунок 1.2 – Система управління навантаженням в ЦОД

Центр обробки даних представляє собою високонавантажену систему. Навіть при великих навантаженнях, центри обробки даних повинні забезпечувати якість надання послуг на високому рівні. Для підтримання якості надання послуг для користувачів, центри обробки даних шляхом підвищення швидкості телекомунікаційної мережі та балансуванням навантаження розподіллять навантаження в системі.

При розподіленні ресурсів в центрах обробки даних, можна виділити наступні модулі, які відповідатимуть за розподілення навантаження в системі управління:

– диспетчер запитів;
– модуль управління.

Про високонавантажені сервіси та керування ресурсами описано в роботах [20-21].

ВИСНОВКИ ДО РОЗДІЛУ 1

Архітектура центрів обробки даних представляє собою високонавантажену систему. Для того, щоб керувати центрами обробки даних потрібно мати систему управління, яка буде правильно розподіляти навантаження між серверами для надання якісного обслуговування користувачам.

При використанні віртуалізації можна обійти обмеженість сучасних серверів, які призначені тільки для запуску однієї операційної системи. Як результат, при використанні віртуалізації, центри обробки даних можуть розгортати велику кількість серверів та віртуальних машин.

Більшість серверів використовують приблизно 20% своєї потужності, а це призводить до збільшення використання кількості серверів в центрах обробки даних. За допомогою віртуалізації можна вирішити проблеми з неефективним розподіленням ресурсів та дозволити декільком ОС працювати використовуючи при цьому один фізичний сервер. Кожна віртуальна машина має доступ до ресурсів основного сервера.

Для ефективного розподілення ресурсів в центрах обробки даних, необхідна наявність системи управління, основна задача якої – керування ресурсами.

2 ПОСТАНОВКА ПРОБЛЕМИ УПРАВЛІННЯ РЕСУРСАМИ ВИСОКОНАВАНТАЖЕНИХ СЕРВІСІВ В ІТ – ІНФРАСТРУКТУРІ

Центри обробки даних представляють собою велику інфраструктуру, яка призначена для розміщення серверних і мережевих систем.

Збільшення запитів користувачів до систем призводить до погіршення якості наданих ІТ-послуг. В цьому випадку для підтримання якості послуг на належному рівні, система управління використовуючи методи масштабування ресурсів та перерозподілу, додає відповідним додаткам або віртуальним машинам додаткові об’єми обчислювальних ресурсів. Таким чином, не відбувається погіршення якості послуг наданих користувачам навіть в години пік.

Основне завдання даної роботи полягає в розроблені методів для управління ресурсами та керування якістю надання послуг для користувачів високонавантажених сервісів в ІТ-інфраструктурі з підвищенням ефективності використання ресурсів в центрах обробки даних. При цьому використовуються розподіл навантаження та збільшення потужності додатків шляхом масштабування обчислювальних ресурсів.

3 РОЗРОБКА МОДЕЛЕЙ ТА МЕТОДІВ ДЛЯ КЕРУВАННЯ РОЗПОДІЛОМ РЕСУРСІВ

3.1 Модель управління ресурсами та навантаженням

Розробка моделі для системи управління ресурсами у високонавантажених системах потребує визначення, що таке віртуальна машина, сервер та навантаження.

1) ЦОД представляє собою спеціалізований майданчик, який слугує для обслуговування серверного обладнання. Позначимо ЦОД, як *Q*.

2) Нехай в центрі обробки даних знаходяться деяка множина фізичних серверів *N* (3.1):

$$N = \{N_1, ..., N_n\}, \tag{3.1}$$

Де:

– n – загальна кількість серверів

– N_i – сервер, який знаходиться в множині фізичних серверів *N* та в дата-центрі *Q*.

Кожен сервер *N*, що розташований в дата-центрі має параметри, які представляють собою характеристику ресурсів:

– Ω_i — процесорна ємність сервера N_i;

– Γ_i — об’єм оперативної пам’яті сервера N_i.

3) сервіси, що розташовані на серверах позначимо, як *A*. Кількість таких додатків на сервері буде дорівнювати *m*. Виходячи з цього можна визначити множину сервісів, що розташована на сервері:

$$A = \{A_1, ..., A_m\}, \tag{3.2}$$

де A_i – сервіс, який знаходиться в множині сервісів А.

4) для кожного сервісу A_j, $j = \overline{1,m}$ маємо параметри:

– ω_j — процесорна ємність сервера, де розташовані екземпляри сервісів A_j;

– γ_j — оперативна пам’ять, де розташовані екземпляри сервісів A_j.

Задачу, що зв'язана з плануванням ресурсів можна розглядати як матрицю $X = \|x_{ji}\|$, яка може прив’язати додатки до серверів.

3.2 Обмеження віртуальних машин

При розміщенні віртуальних машин на серверах повинні враховуватись наступні обмеження:

1) характеристики всіх застосувань не повинні перевищувати параметри сервера;

2) на основі навантаження, що створюють користувачі повинні бути визначені вимоги до серверів. Сервери повинні обслуговувати користувачів без втрати якості надання послуг.

3.3 Розподіл ресурсів в центрах обробки даних

Основна ціль при управлінні ресурсами – розподілу ресурсів базуючись на потребах сервісів та ресурсах серверів.

Обмеження 3.3 потребує, щоб умови були виконані для користувачів на сервісах A_j, на основі ресурсів:

$$\sum_{j=1}^{m} x_{ji} \cdot \gamma_j \le \Gamma_i, i = \overline{1,n}. \tag{3.3}$$

Таким чином, формула 3.4 представляє собою обмеження оперативної пам’яті при розміщенні сервісів на серверах:

$$\sum_{j=1}^{m} x_{ji} \cdot \omega_j \le \Omega_i, i = \overline{1,n}. \tag{3.4}$$

Основною задачею хостингових компаній є надання якісних послуг та задоволення потреб користувачів. Аналіз навантаження під час роботи системи слугує для контролю якості надання послуг. Дані, що були отримані після аналізу в подальшому можуть бути використані для здійснення розподілення ресурсів на

серверах, при підвищенні навантаження до критичних значень. Це допоможе підтримувати якість послуг для користувачів на достатньому рівні.

3.4 Модель керування ресурсами

Хостинг – це послуга надання ресурсів на серверах. За допомогою хостингу можна розміщати файли та інформацію на сайті на якому є виконуване програмне забезпечення (ПЗ). Здебільшого, хостинг бере на себе відповідальність в обслуговуванні користувачів та надані файлового сховища на окремому сервері. Додатковими послугами для хостингу можуть бути: підтримка працездатності та балансування навантаження ресурсів.

Кожна хостингова компанія повинна задовільнити ряд потреб для розподілення ресурсів в центрах обробки даних, а саме:

1) високопродуктивність – системи для яких важлива швидкість виконання запитів до сервісів. Саме для швидкості виконання запитів застосовується розподілення ресурсів. Розподілення ресурсів допомагає забезпечити сервіси необхідними ресурсами для підтримання якісного обслуговування користувачів. При розподілені, також, необхідно враховувати обчислювальні потужності, які є в наявності у ЦОД. Обчислювальні ресурси дата-центру можна формалізувати у вигляді обмеження, наведеного у формулі 3.5:

$$\sum_{j=1}^{m} x_{ji} \cdot \omega_j \le \Omega_i, i = \overline{1,n}. \tag{3.5}$$

2) високонавантаженість – системи (сервіси), які можуть обслуговувати велику кількість користувачів без втрати якості обслуговування.

При моделюванні системи, потрібно враховувати кількість ядер та оперативну пам’ять, яка використовується для обслуговування додатків та користувачів в центрах обробки даних. За допомогою моніторингу характеристик віртуальних машин та серверів, буде здійснюватися контроль операцій введення та виведення й можливості доступності сервісу.

Визначимо змінні, для відображення завантаження на серверах:

$K_{ОС}$ – навантаження на оперативну пам’ять сервера при виконанні операцій, що обслуговують користувачів;

$K_{доп}$ – навантаження оперативної пам’яті, яка обмежується ресурсами на серверах;

F – частота тактів або операцій для користувача;

C – технічна можливість розгортання додатка в системі.

M – кількість ядер в системі.

Тепер можемо сформулювати формулу для знаходження навантаження на серверах при розподілі ресурсів у центрах обробки даних (3.6):

$$L(N) = K_{ОС} + K_{доп} + \frac{(C * N)}{(F * M)} \tag{3.6}$$

3.5 Модель розподілу ресурсів на серверах

Перед тим, як визначити методи вирішення проблем з навантаженням потрібно чітко визначити, як саме буде розподілятись навантаження.

Виходячи з цього весь ресурс сервера визначатиметься формулою 3.7:

$$\sum_{K=1}^{R} C_k, \tag{3.7}$$

де:

– $k = \overline{1,R}$;

– C_k – технічна можливість розгортання додатка в системі;

– R – можливість розгортання сервісу на визначеному сервері.

Для того, щоб правильно розподілити ресурси для віртуальної машини при навантажені їй потрібна кількість ресурсів r_i.

Загальну кількість ресурсів, що використовуються на всіх віртуальних машинах можна знайти використовуючи формулу 3.8:

$$A_i = \sum_{k=1}^{R} r_k \tag{3.8}$$

Для знаходження всіх віртуальних машин на сервері, можна використати формулу 3.9:

$$N = \frac{\sum_{k=1}^{R} C_k}{\sum_{k=1}^{R} r_k} \tag{3.9}$$

Стан віртуальної машини на сервері характеризується величною $x_i(t)$. Характер поведінки віртуальної машини, буде таким:

– $x_i(t) = 1$ – машина працює на сервері;

– $x_i(t) = 0$ – не працює і не споживає ресурсів.

Тоді, кількість запущених віртуальних машин на сервері можна знайти використовуючи формулу 3.10:

$$\sum_{i=1}^{N} x_i(t) \tag{3.10}$$

Враховуючи 3.10, обмеження ресурсів сервера прийме вигляд 3.11:

$$\sum_{i=1}^{N} x_i(t) \sum_{k=1}^{R} r_k \le \sum_{k=1}^{R} C_k , \tag{3.11}$$

де $i = \overline{1, N}$, $k = \overline{1, R}$.

Розподілення навантаження на серверах може бути різним, таким чином споживання ресурсів в центрах обробки даних для віртуальних машин теж відрізняється. Під час перерозподілу ресурсів потрібно враховувати, скільки віртуальна машина використовує ресурсів та як її краще розмістити в центрі обробки даних, щоб гарантувати якість обслуговування користувача в системі. Опис розподілу ресурсів описано в роботі [22].

3.6 Методи управління ресурсами

Проаналізувавши проблему розподілення ресурсів у високонавантажених системах було розроблено декілька методів для покращення розподілу ресурсів.

Для реалізації методів розподілу ресурсів, система повинна проводити моніторинг компонентів IT-інфраструктури таких як сервери, пам’ять, процесор, здійснювати контроль операцій введення та виведення. Для цього необхідно впровадити моніторинг системи.

Системи моніторингу в системі управління можуть слідкувати за часом виконання запитів або критичними піками навантаження, чи кількістю повідомлень у системі, які потрібно обробити.

Для вирішення проблеми з розподіленням навантаження у високонавантажених системах для кожного з методів потрібно обрати певний ряд характеристик системи, які будуть використані алгоритмами для аналізу. На їх основі система управління генеруватиме команди для правильного розподілення навантаження. Наведемо метрики, які використовуються методами розподілення навантаження:

- затримка трафіку;
- відсоток росту навантаження в системі;
- аналіз навантаження в певні проміжки часу;
- кількість повідомлень в черзі для обробки системи.

Як показав Теленик та співавтори у [23], основна проблема при розподілі ресурсів у ЦОД полягає у затримці трафіку, який проходить через систему та відсоток росту навантаження в системі. Саме ці дві метрики обрані для вирішення проблеми забезпечення якісного обслуговування. В даній роботі будуть розроблені та представлені методи вирішення проблеми розподілення навантаження, що базуються на дисперсії та рекурентному співвідношенні. Методи розподілення ресурсів допоможуть системі керувати якістю надання послуг для користувачів високонавантажених сервісів в IT-інфраструктурі з підвищенням ефективності використання ресурсів в центрах обробки даних.

3.7 Вирішення задачі розподілу ресурсів високонавантажених сервісів з використанням дисперсії

Дисперсія має властивість виміру мінливості, використовуючи середнє значення та так звану волатильність. Волатильність представляє собою ризик або ризиковий показник, який можна використовувати для управління ресурсами.

При обчисленні дисперсії (3.12) для набору чисел, потрібно враховувати, що дисперсія дорівнює нулю тільки в одному випадку – коли всі числа які були в наборі є нулями.

$$s^2 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}, s = \sqrt{x^2}, \quad (3.12)$$

де $\bar{x}$ представляє собою середнє значення з вибірки (3.13):

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (3.13)$$

Кроки, що визначенні у формулі 3.12 можна описати за допомогою двох етапів:

1) знайти середнє значення в першому проході;

2) знайти квадрат відмінності від середнього значення, який був порахований на першому кроці.

Щоб підрахувати дисперсію потрібно використати формули 3.12 та 3.13. Тоді, формула, знаходження дисперсії, матиме вигляд 3.14:

$$s^2 = \frac{\sum_{i=1}^{N}(x^{-2} - 2\bar{x}x_i + \bar{x}^2)}{N-1} = \frac{\sum {x_i}^2 - 2N\overline{x^2}}{N-1} = \frac{\sum {x_i}^2 - N\overline{x^2}}{N-1} \quad (3.14)$$

Після того, як була визначена формула для знаходження дисперсії, можна виділи основні кроки алгоритму:

1) взяти вибірку даних в якій треба знайти дисперсію;

2) скористатись формулою 3.12 за допомогою неї можна зрозуміти як вибірка даних, буде розподілятись по точкам даних. Якщо ж дисперсія буде наближена до нуля, то дані будуть більше згруповані один біля одного;

3) знайти середнє значення вибірки – означає, що потрібно додати всі точки даних разом, а потім поділити на кількість точок даних;

4) відняти середнє значення від кожної точки даних, за допомогою цього можна знайти відхилення числа від середнього значення і на скільки саме;

5) піднести до квадрату кожен результат з п.4, де середнє значення повинно бути рівнем нулю, це говорить про те чи правильно пораховані данні. Для того, щоб дані не були від'ємними потрібно знайти квадрат для кожного відхилення. Після цього всі числа будуть позитивними;

6) знайти суму квадратів всіх чисел з п.5;

7) розділити на N-1, де N – кількість точок, що існувала у вибірці. Після цього кроку отримаємо середнє значення квадратного відхилення, яке може бути узгодженим з дисперсією цього зразка.

8) проаналізувати дисперсію та стандартне відхилення.

Після опису знаходження дисперсії, можна описати кроки для розподілення ресурсів та навантаження з використанням дисперсії. Алгоритм дисперсії наведено в додатку В.

Алгоритм для розподілення ресурсів з використанням дисперсії слугує для того, щоб розподілити ресурси базуючись на часі виконання запитів системи. Якщо час виконання запитів є далеким від середнього в наборі всіх запитів, то система починає повільно опрацьовувати запити та починає погіршуватись якість надання послуг для користувача. В такому випадку, потрібно розподілити ресурси. Для вирішення цієї проблеми є два варіанти – збільшити або зменшити ресурси, щоб волатильність в початкових даних зменшилась.

Приклад застосування дисперсії на основі кількості запитів та часу їх виконання зображено на рис. 3.1.

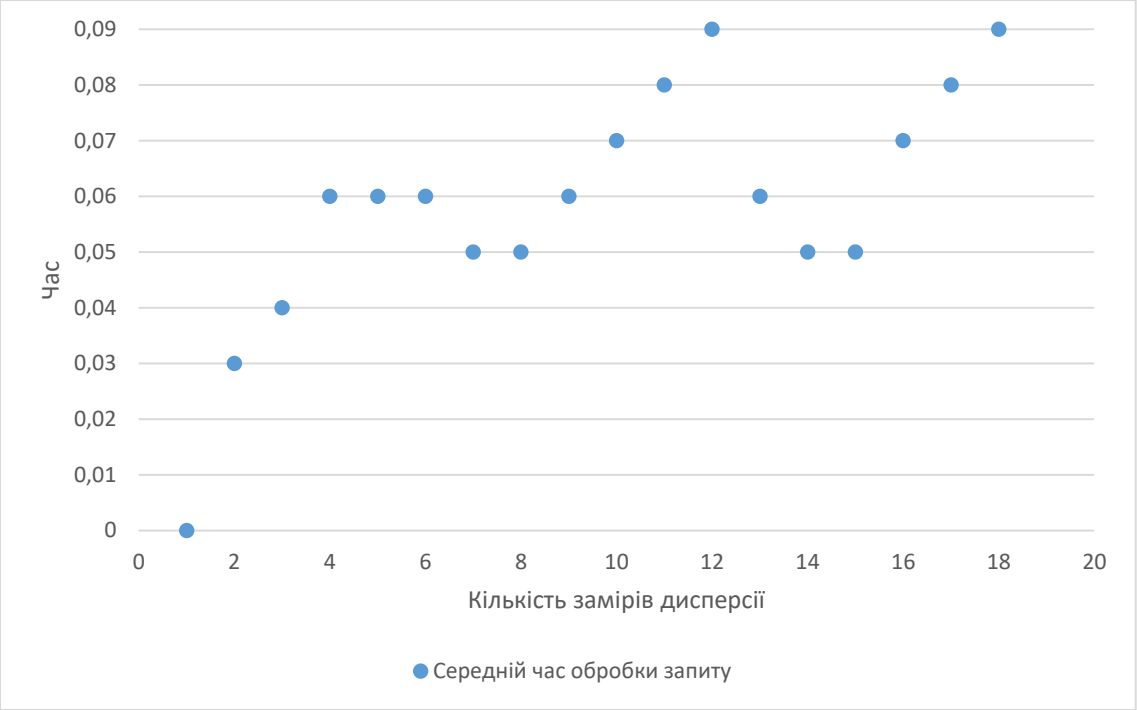


Рисунок 3.1 – Розподілення запитів в системі управління основуючись на дисперсії

3.8 Вирішення задачі розподілу ресурсів високонавантажених сервісів рекурентним методом

Для вирішення проблеми розподілу ресурсів можна передбачати, що буде з ресурсами через якийсь певний час. Які види ресурсів та в якій кількості потрібні для обслуговування та надання якісного сервісу користувачам. Одним із методів для вирішення задачі з розподілу ресурсів є зворотне співвідношення. За допомогою зворотнього співвідношення можна аналізувати дані та здійснювати вплив на систему при досягненні граничних допустимих характеристик, що задані як критичні границі в системі управління. Також, зворотні співвідношення використовуються в аналізі алгоритмів для того, щоб отримати асимптотичні оцінки рекурсивних співвідношень.

Нехай маємо послідовність $(a_0, a_1, a_2, a_3, \dots)$, яка задовольняє деякому рекурентному співвідношенню. Потрібно отримати вираз a_n при n, де $n \geq 0$ в замкнутому вигляді. Якщо ж можна буде знайти функцію послідовності, то для

знаходження рекурентного співвідношення можна використовувати один і той самий алгоритм.

Зобразимо лінійне однорідне рекурентне співвідношення 2-го порядку з довільним постійним коефіцієнтом 3.14:

$$\begin{cases} 0, n = 0 \\ 1, n = 1 \\ F_{n-1} + F_{n-2}, n \geq 2 \end{cases} \quad (3.14)$$

Порядок співвідношення представляє глибину, тобто кількість попередніх елементів, які необхідні для того, щоб обчислити елементи з номером n.

Після цього, потрібно знайти функцію послідовності 3.15:

$$G(z) = \sum_{n=0}^{\infty} a_n z^n = a_0 + a_1 z + a_2 z + \cdots \quad (3.15)$$

Після того, як були визначені основні формули можна записати алгоритм рекурентного співвідношення.

Алгоритм для рекурентного співвідношення:

1) записати рекурентне співвідношення і початкові дані, які будуть знайдені за формулою 3.15;

2) помножити кожен рядок на z і підсумувати рядки всіх $n \geq 0$;

3) в отриманому рівнянні 3.15 привести всі суми $\sum n$ до замкнутого виду. Отримати рівняння;

4) знайти $G(z)$ в явному вигляді (розв'язати рівняння) і розкласти генератристичну функцію в ряд за ступенем z.

В залежності від значення дисперсії обчисленого за формулою 3.14 ресурси будуть збільшуватись або зменшуватись, що дозволить підтримувати якість обсуговування користувачів на налжному рівні.

ВИСНОВКИ ДО РОЗДІЛУ 3

Після проведення аналізу проблеми розподілення ресурсів у високонавантажених сервісах, було розроблено два методи для управління ресурсами та якістю надання послуг для користувачів високонавантажених сервісів в IT-інфраструктурі з підвищенням ефективності використання ресурсів в центрах обробки даних.

Для реалізації методів ефективного використання ресурсів, системі управління потрібно надати інформацію про поведінку системи протягом минулих n-днів чи m-місяців, для цього потрібно зібрати метрики системи. Метрики представляють собою основну інформацію про навантаження в системі.

Забезпечення керування якістю надання послуг для користувачів високонавантажених сервісів в центрах обробки даних може відбуватися за допомогою двох методів:

- дисперсійний метод – алгоритм по розподіленню ресурсів з використанням дисперсії, слугує для того щоб правильно розподілити ресурси базуючись на часі виконання запитів системи;
- рекурентний метод допомагає передбачати, що буде з ресурсами, коли навантаження буде зростати. Базуючись на граничних умовах, що задані системою управління.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ РОЗПОДІЛУ РЕСУРСІВ ВИСОКОНАВАНТАЖЕНИХ СЕРВІСІВ У IT – ІНФРАСТРУКТУРІ

4.1 Особливість мови програмування C#

Для реалізації запропонованих методів розподілу ресурсів високонавантажених сервісів у IT-інфраструктурі була обрана мова програмування C# та інтегроване середовище розробки Microsoft Visual Studio 2017.

Використовуючи мову C# для розроблення програмного забезпечення, розробники отримають можливість використовувати особливості мови: автоматичне збирання сміття, статичні методи, перевантаженні оператори та об’єктне орієнтоване програмування. Тільки автоматичне збирання сміття дозволяє розробнику не звертати увагу на виділення та звільнення ресурсів після їх використання. Таким чином, автоматичне збирання сміття забезпечує розробника від можливості витіку пам’яті, як це часто буває в інших мовах програмування.

Для розробки графічного інтерфейсу була обрана технологія Windows Presentation Foundation (WPF). Вона дозволяє розробляти графічні будь-якої складності, що надає розробникам широкі можливості для візуалізації інформації. Серверна частина додатку реалізована на платформі .NET мовою C#. Розробка здійснювалася з врахуванням принципів предметно-орієнтованого проектування та з використанням шаблонів проектування програмного забезпечення

4.2 Особливості платформи ASP.NET Core

ASP.NET Core – технологія з відкритим вихідним кодом для швидкої розробки веб-додатків розроблена компанією Microsoft. На відміну від класичної платформи ASP.NET, ASP.NET Core є крос-платформною, що дозволяє розгортати додатки на різних платформах, серед яких: Linux, Mac OS, Windows.

Порівняємо ASP.NET Core з прямим конкурентом, платформою Java. Переваги ASP.NET Core над платформою Java;

1) Середовище виконання CLR, яке дозволяє використовувати мови програмування високого рівня, зі зручним синтаксисом (C#, F#);

2) відкритий вихідний код та значна спільнота, яка підтримує проект;

3) немає обмежень щодо платформи на якій розгортаються додатки.

4.3 Принципи предметно-орієнтованого проектування

Розробка програмного забезпечення з використанням предметно-орієнтованого проектування починається з визначення предметної області з представниками замовника чи бізнес-аналітиками. Саме взаємодія розробників та замовника є ключовим аспектом даного підходу.

У процесі взаємодії розробники краще пізнають предметну область і поступово моделюють її в програмному коді на основі ряду базових абстракцій, а саме:

– клас-значення, об’єкт, який не може бути змінений після створення. Якщо потрібно щось змінити в цьому об’єкті, то потрібно створювати новий. Таких об’єктів може бути багато і як правило вони використовуються у поєднанні з класами-сутностями(Entity).

– сутність(entity), описує поведінку певної сутності з предметної області, зазвичай агрегує декілька класів-значень та операції які можна здійснювати над ними.

– агрегат - клас який може об’єднувати всі інші види об’єктів. Застосовується для опису меж бізнес транзакцій.

4.4 Об'єктна модель системи розподілення ресурсів

Об’єктна модель описує структуру об’єктів, що входять в систему, її атрибути, операції, взаємозв’язки з іншими об’єктами. В об’єктній моделі повинні бути зображені поняття і об’єкти додатку, які важливі для розроблювальної системи. Об’єктом в об’єктній моделі будемо називати поняття, абстракцію чи будь-яку річ з чітко визначеними межами, що описана в контексті даної прикладної проблеми.

Після проведення аналізу предметної області для проблеми розподілу ресурсів у центрах обробки даних, була розроблена об’єктна модель яка складається з таких об’єктів:

1) VM;
2) Server;
3) DC;
4) VarianceAlgorithm;
5) RecurrenceAlgorithm.
6) IHardware;
7) IEquatable.
8) Entity;
9) ValueObject.
10) Data Generator
11) PerfomanceMetric;
12) Logger.

Всі об’єкти одного і того ж класу характеризуються однаковим набором властивостей або атрибутів. Об’єднання об’єктів в класи визначається не набором властивостей, а семантикою і називаються такі об’єкти агрегатами.

4.5 Інтерфейс IEquatable

Інтерфейс IEquatable описує операцію визначення рівності двох об’єктів. Діаграми інтерфейсу IEquatable та методу Equals неведені на рис. 4.1.

Рисунок 4.1 – Діаграма інтерфейса IEquatable

Інтерфейс IEquatable декларує тільки один метод, як видно з рис. 4.1:

– Equals основна задача методу визначити чи є два екземпляри класів рівні між собою.

4.6 Базовий клас Entity

Клас Entity має ідентифікатор за допомогою якого можна знайти цей об’єкт в базі даних.

Два об’єкти класу Entity в додатку можна вважати рівними тоді і тільки тоді, коли ідентифікатори цих об’єктів є рівними між собою.

Клас Entity допомагає керувати іншими об’єктами в системі та слугує класом для обміну інформацією між базою даних та клієнтами. Entity можна використовувати для керування іншими об’єктами, такими як об’єкти-значення (ValueObject).

В класі Entity реалізований один метод: Equals, який реалізує операцію еквівалентності двох екземплярів даного класу за ідентифікаторами.

Діаграма класу Entity наведена на рисунку 4.2.

Рисунок 4.2 – Діаграма класу Entity

Згідно рисунку 4.2 клас Entity реалізує інтерфейс IEquatable. Базовий клас Entity далі використовуватимемо для реалізації об’єктів, які будуть зберігатися у базі даних, та матимуть ідентифікатор.

Повна ієрархія наслідування класу Entity наведена на рис. 4.3.

Рисунок 4.3 – Діаграма класів ієрархії класу Entity

4.7 Незмінний клас ValueObject

Клас ValueObject описує незмінний тип. На відмінну від сутності, унікальність класу ValueObject визначається всіма властивостями цього об’єкту.

Незмінним вважається тип який, не може змінюватися після створення.

Будь який метод, який повинен змінити стан об’єкту-значення повинен повертати новий об’єкт, а не модифікувати існуючий об’єкт. Діаграму класу ValueObject наведено на рисунку 4.4.

Рисунок 4.4 – Діаграма класу ValueObject

Наведемо опис методів описаних в класі ValueObject:

– Equals основна задача метода знайти чи є два екземпляра рівні між собою за допомогою зрівняння всіх атрибутів класу;

– EqualsCore слугує для того, щоб зрівнювати два екземпляра класу по базовому методу Equals. Не може бути перевантажений;

– GetHashCode обчислює хеш-код об’єкта та визначає;

– GetHashCodeCore слугує для отримання хеш-коду з базового класу.

4.8 Базовий інтерфейс IHardware

IHardware — інтерфейс використовується для того, щоб описати характеристики обладнання.

Діаграма інтерфейсу IHardware наведена на рисунку 4.5.

Рисунок 4.5 – Діаграма інтерфейсу апаратного засобу IHardware

Як можна бачити з рис 4.5, інтерфейс IHardware описує 2 атрибути:

– CPU – процесорна ємність для обладнання;

– RAM – ємність оперативної пам’яті для обладнання.

4.9 Класу VirtualMachine

Клас VirtualMachine описує поведінку віртуальної машини в центрі обробки даних. Діаграма класу VirtualMachine наведена на рисунку 4.6.

Рисунок 4.6 – Діаграма класу віртуальна машина

Діаграма класів об’єкту VirtualMachine зображена на рис. 4.7.

Рисунок 4.7 – Діаграма класів віртуальної машини

З рис. 4.7 можна бачити, що VirtualMachine унаслідується від базового класу Entity.

Клас VirtualMachine має набір операцій:

1) Copy – це операція копіювання, яка створює такий же самий об’єкт з такими властивостями;

2) VirtualMachine – конструктор, для створення об’єкту, який здійснює перевірку параметрів, які потрібні для створення об’єкту.

4.10 Об’єкт класу Server

Клас Server – описує множину об’єктів для керування віртуальними машинами в центрі обробки даних. Клас Server розширяє базовий клас Entity та реалізує інтерфейс IHardware..

Основні операції класу Server:

1) Add - операція створення сервера в дата центрах;

2) Delete - операція видалення сервера, якщо він вже не потрібний;

3) Edit – операція редагування властивостей сервера, дозволяє змінити ім’я сервера або змінити дата центр, на якому розміщений даний сервер.

Діаграма класу Server з усіма атрибутами зображена на рис. 4.8.

Рисунок 4.8 – Діаграма класу Server

Опишемо операції класу Server, які пов’язані з обслуговуванням віртуальних машин:

1) IsServerFully() – операція повертає правдиве чи не правдиве твердження про те, чи є сервер повними чи ні;

2) IsServerInclude() - функція дозволяє перевіряти чи розгорнута віртуальна машина на цьому сервері, повертає правдиве чи не правдиве твердження;

3) Конструктор слугує для того, щоб створити об’єкт з тими властивостям, які потрібні саме цьому серверу. Також реалізує валідацію даних.

На діаграмі класів рис. 4.9, можна бачити всі методи, що використовуються у класі Server.

Рисунок 4.9 – Діаграма класів серверу

З рис. 4.9 можна бачити, що клас Server описує множину об’єктів, які інкапсулюють логіку керування серверами в середині дата-центрів та визначає основні характеристики:

- Ω_i – процесорна місткість сервера N_i, яка визначена в інтерфейсі IHardware як CPU;
- Γ_i – об’єм оперативної пам’яті сервера N_i, була визначена в інтерфейсі IHardware, як RAM;

4.11 Об’єкт класу DataCenter

Клас DataCenter відповідає за розподілення віртуальних машин та серверів на основі характеристик в центрах обробки даних. DataCenter розширяє базовий клас Entity. Діаграму класу DataCenter зображена на рисунку 4.10.

Рисунок 4.10 – Діаграма класу DataCenter

З рис. 4.11 можна бачити, що DataCenter розширяє клас Entity.

Рисунок 4.11 – Діаграма класів DataCenter

Клас DataCenter слугує для розподілення серверів в середині центру обробки даних та допомагає керувати іншими об’єктами в системі. Основні методи, визначені на рис. 4.10:

1) створення сервера, якщо сервери не справляються з навантаженням;
2) видалення сервера, якщо навантаження вже немає;
3) редагування сервера.
4) конструктор, який приймає всі допоміжні атрибути, які необхідні для створення об'єкта.

4.12 Клас Variance

Клас Variance реалізує операцію обчислення дисперсії. Variance потрібен для того, щоб правильно розподілити ресурси на основі часу виконання запитів до системи. Якщо розподіл часу виконання запитів відрізняється від середнього, то система починає повільно опрацьовувати запити, що в свою чергу приводить до погіршення якості надання послуг для користувача. В такому випадку, потрібно розподілити ресурси.

Об’єкт має параметри, які потрібні для розрахунку дисперсії:

1) список серверів;
2) список віртуальних машин.

Діаграма класу Variance, зображена на рисунку 4.12 та 4.13.

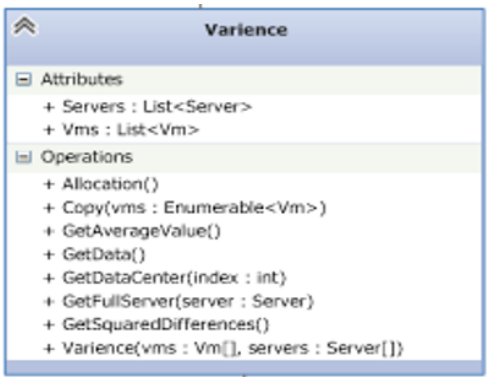


Рисунок 4.12 – Діаграма класу Variance

Клас Variance зображений на рисунку 4.12 визначає основні операції для підрахунку дисперсії на основі часу виконання запитів в середині СУІ:

1) створення об’єкта для алгоритму;

2) видалення віртуальної машини або сервера;

3) редагування віртуальної машини та серверу.

4) операція копіювання потрібна для створення додаткових серверів та віртуальних машини в центрах обробки даних;

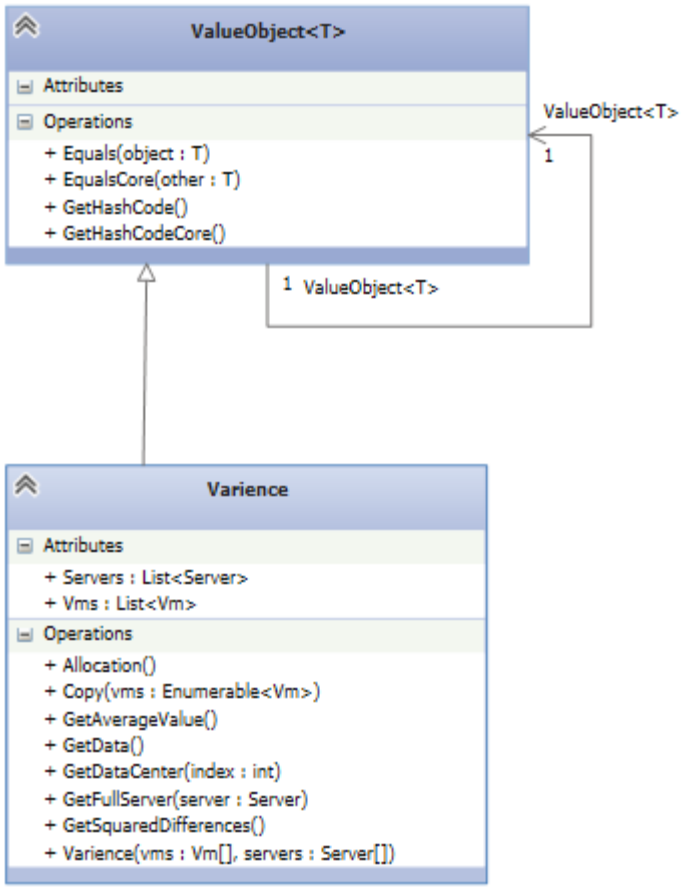


Рисунок 4.13 – Діаграма класів Variance

5) GetDataCenter за допомогою функції можна повністю скомплектувати дата-центр з серверами та віртуальними машинами звертаючи увагу на розподіл ресурсів та аналіз дисперсії;

6) GetFullServer ця функція допомагає нам скомплектувати сервер, який буде мати характеристики, які не будуть виходити за межі ресурсів сервера;

7) GetAverageValue знаходить середнє значення для всіх даних, що були зібрані за період аналізу. Якщо значення дорівнюватиме нулю, – це означає, що дисперсія не має відхилення;

8) GetSquaredDifferences обчислення квадратів різниць середнього значення, та значення величи;

9) GetData за допомогою цієї функції алгоритм дисперсії має можливість отримати данні з бази даних, для того, щоб мати можливість їх проаналізувати та зробити правильне розподілення навантаження;

10) Allocation представляє собою, найголовнішу функцію за допомогою якої можна розподілити ресурси в центрах обробки даних на основі дисперсії, отриманої на попредньому кроці.

4.13 Клас RecurrenceRelation

Клас RecurrenceRelation слугує для розподілення ресурсів в середині ЦОД, за допомогою рекурентного співвідношення. RecurrenceRelation аналізує дані та здійснює вплив на систему за умови досягнення граничних допустимих значень характеристик, які задані системою управління.

Діаграму класу RecurrenceRelation зображено на рис.4.14.

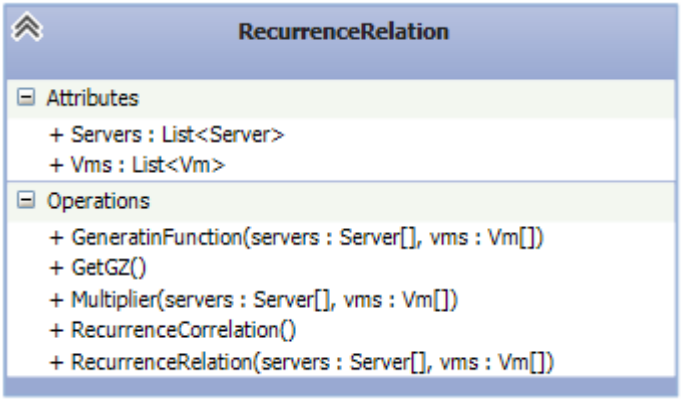


Рисунок 4.14 – Діаграма класу RecurrenceRelation

Клас RecurrenceRelation на рис. 4.14 визначає основні методи, для розподілення ресурсів в середині ЦОД:

1) RecurrenceCorrelation слугує для запису рекурентного співвідношення;

2) Multiplier допомагає підсумувати всі рядки за формулою, що визначена в розділі 3.8;

3) GeneratingFunction для отриманої нерівності приводить суму до замкнутого вигляду;

4) GetGZ допомагає виразити результат в явному вигляді і розкласти функцію в ряд за ступенем z;

Основні модулі на які поділяється система – модуль віртуальних машин, модуль серверів та модуль дата-центрів. Діаграма компонентів навелена в додатку Б2. Кожен модуль містить набір операцій, призначених для взаємодії з модулями.

«Модуль віртуальної машини» відображає операції, що можуть виконуватись над віртуальними машинами в центрах обробки даних, а саме:

– редагування віртуальних машин;

– додавання віртуальних машин;

– видалення віртуальних машин.

«Модуль серверів» слугує для роботи з серверами, та має такий самий набір операцій, як і «Модуль віртуальної машини».

Останній модуль – «Модуль дата-центрів». Він відображає результат розподілення віртуальних машин по серверам всередині дата-центрів.

Діаграма класів всіх об’єктів, що були використані для реалізації розподілу ресурсів СУІ зображена в додатку Е.

ВИСНОВКИ ДО РОЗДІЛУ 4

В даному розділі була описана програмна реалізація методів розподілу ресурсів високонавантажених сервісів в ІТ-інфраструктурі. Описано та обґрунтовано переваги та недоліки мови C#.

Архітектура системи представлена у вигляді діаграми класів, компонентів та алгоритму. Також, було надано специфікацію функцій з детальним описом.

5 ВИПРОБУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Розділ тестування включає в себе тестування, що треба виконати для того, щоб перевірити програмне забезпечення на функціональні вимоги, що були описані в технічному завдані.

5.1 Мета тестування

Мета тестування програмного забезпечення полягає у виявленні розбіжностей розробленого продукту з вимогами, що були складенні на етапі проектування системи.

5.2 Загальні положення

Нефункціональні характеристики слугують для тестування програмного забезпечення, що описані як нефункціональні вимоги до продукту. В свою чергу тестування поділяється на:

- функціональне;
- нефункціональне;
- структурне тестування;
- регресивне тестування;

5.2.1 Функціональне тестування

Функціональне характеристики слугують для тестування програмного забезпечення, що описані як функціональні вимоги до продукту.

Для здійснення тестування необхідно провести підготовку системи. Наведемо перелік кроків необхідно виконати:

1) підготувати тестові данні;
2) врахувати вимоги до бізнесу;
3) підготувати еталонні вихідні дані, для їх подальшого порівняння з вихідними даними отриманні в результаті роботи системи;

4) порівняти вихідні дані отримані в результаті роботи системи з еталоном підготовленим у пункті 3.

Основне завдання функціонального тестування - відтворити дії користувача та виявити розбіжності в поведінці системи та вимогами описаними в документації до програмного забезпечення. Проте, цей підхід має ряд недоліків – тестувальники можуть пропустити деякі вимоги до системи або навпаки – інтерпретувати їх зовсім по-іншому

5.2.2 Нефункціональне тестування

Нефункціональне тестування може слугувати вимогам, що не описані в документах, наприклад: швидкість опрацювання запитів та надійність системи.

Нефункціональне тестування, як правило, використовується після проведення функціонального тестування.

5.2.3 Регресивне тестування

Регресивне тестування використовується, як останній крок для передачі програмного забезпечення замовнику. Регресивне тестування слугує для знаходження всіх помилок в додатку.

5.3 Результати тестувань

Метою створення програмного забезпечення є надання можливості розподілення ресурсів у високонавантаженій IT-інфраструктурі. Вона дозволяє контролювати ресурси та розподіляти в залежності від завантаженості обладнання в ЦОД.

Опишемо функціональність, яка реалізована у розробленому додатку. Перша реалізована функція, – створення віртуальних машин та серверів. Після цього користувач може здійснити розподілення ресурсів в серверах та дата – центрах за допомогою різноманітних методів та підібрати найкращу конфігурацію

роботи ЦОД. Тест функціональності додавання віртуальної машини наведемо в таблиці 5.1.

Таблиця 5.1 – Тест додавання віртуальної машини

Ціль тесту	Перевірка функції «Додавання віртуальних машин»
Початковий стан	Додаток запущений
Вхідні дані	Відкрита перша вкладка в додатку
Проведення тестування	1) Заповнені всі поля (рисунок 5.1): – назва віртуальної машини; – розмір оперативниї пам'яті; – розмір процесора. 2) Нажати кнопку “Додавання”
Очікуванні результати	Віртуальна машина була додана в систему, з таким самим ім’ям та описом, який був заданий користувачем.
Стан після проведених випробувань	Віртуальна машина була додана в систему, з таким самим ім’ям та описом, який був заданий користувачем. Відображено результат роботи на рисунку 5.2.

Рисунок 5.1 – Заповнення полів для створення віртуальної машини

При заповнені атрибутів для віртуальних машин, серверів та дата-центрів спрацьовує підсистема валідації, яка перевіряє значення введених параметрів на правильність.

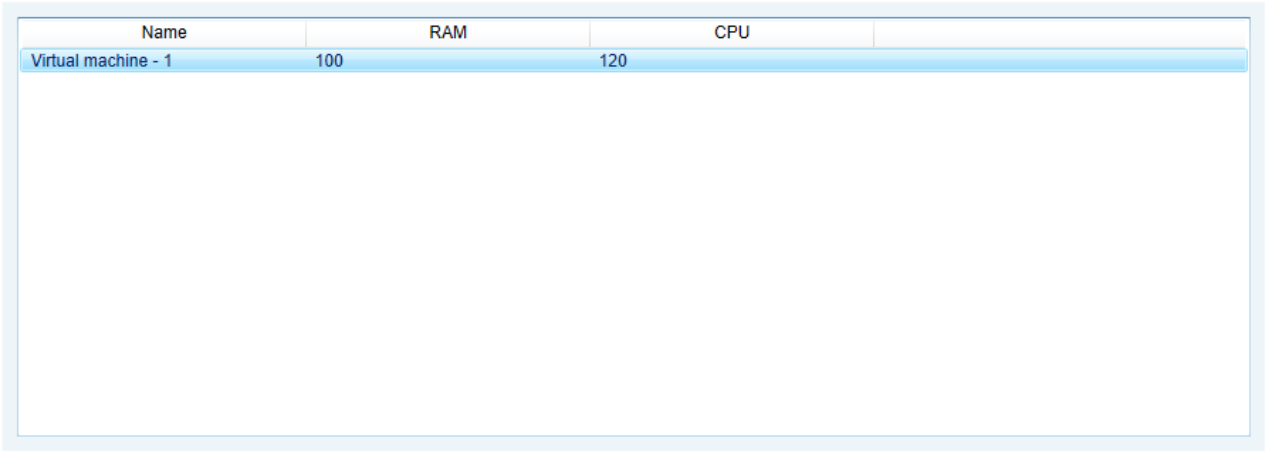


Рисунок 5.2 – Очікуваний результат після додавання віртуальної машини

Далі наведемо опис тесту функціональності додавання серверів (наведено в таблиці 5.2).

Таблиця 5.2 – Тест додавання серверів

Ціль тесту	Перевірка функції «Додавання серверів»
Початковий стан	Додаток запущений
Вхідні дані	Відкрита друга вкладка в додатку
Проведення тестування	3) Заповнені всі поля (рисунок 5.3): – ім'я сервера; – розмір оперативної пам'яті; – розмір процесора. 4) Натиснути кнопку “Додавання”
Очікуванні результати	Сервер був доданий в систему. В списку серверів відображається назва, та опис, який був вказаний при створенні компонента.
Стан після проведених випробувань	Сервер був доданий в систему. В списку серверів відображається назва, та опис, який був вказаний при створенні компонента. Результат роботи наведено на рисунку 5.4.

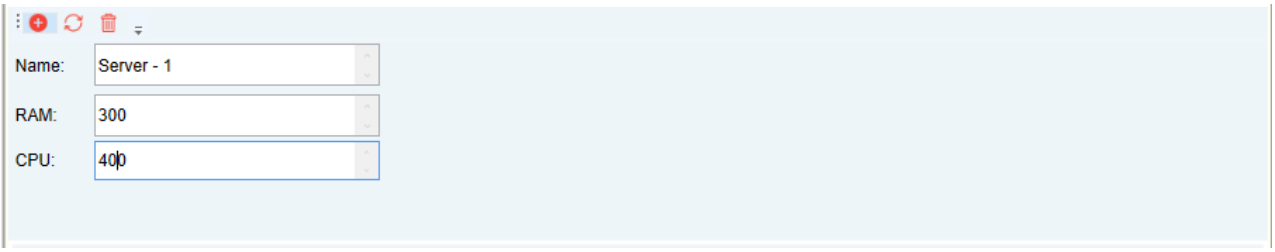


Рисунок 5.3 – Заповнення полів для створення сервера

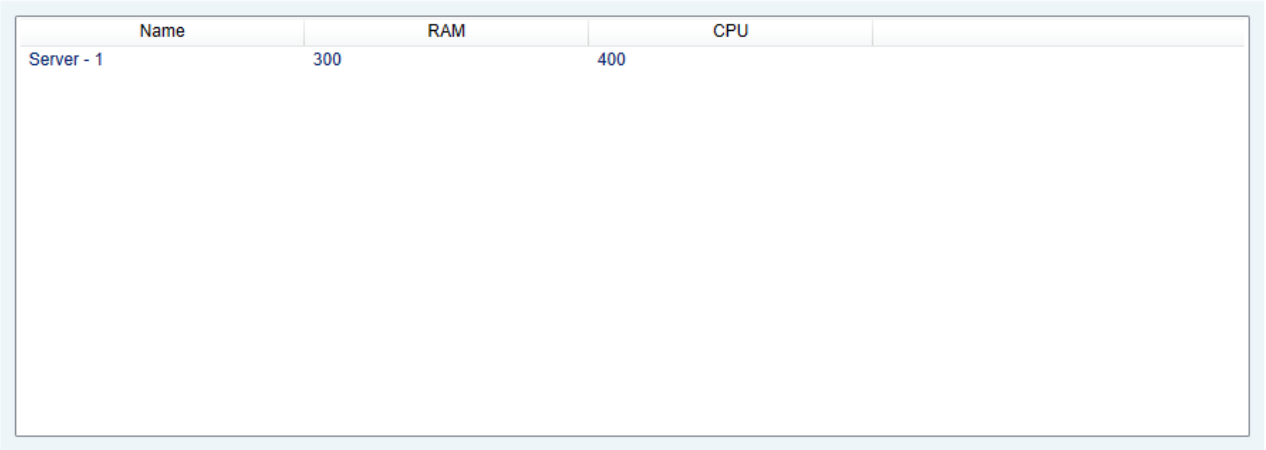


Рисунок 5.4 – Очікуваний результат після додавання сервера

Далі наведемо опис тесту отримання результатів щодо розподілення віртуальних машин на серверах в середині дата – центру (наведено в таблиці 5.3).

Таблиця 5.3 – Тест розміщення віртуальних машин на сервері

Ціль тесту	Перевірка функції «Розміщення віртуальних машин в сервері»
Початковий стан	Додаток запущений
Вхідні дані	Відкрита вкладка “Розміщення віртуальних машин на серверах”.
Проведення тестування	Вибрати один із серверів (рисунок 5.5).
Очікуванні результати	Після вибору сервера користувачі повинні отримати список віртуальних машин, що є в даному сервері.

Продовження таблиці 5.3

Стан після проведених випробувань	Після вибору сервера користувачі повинні отримати список віртуальних машин, що є в даному сервері. Відображено результат роботи на рисунку 10.6.

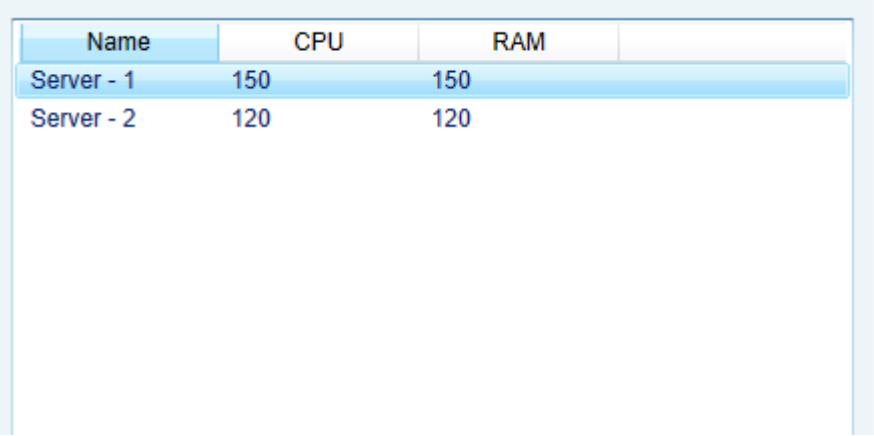


Рисунок 5.5 – Вибір серверу зі списку серверів

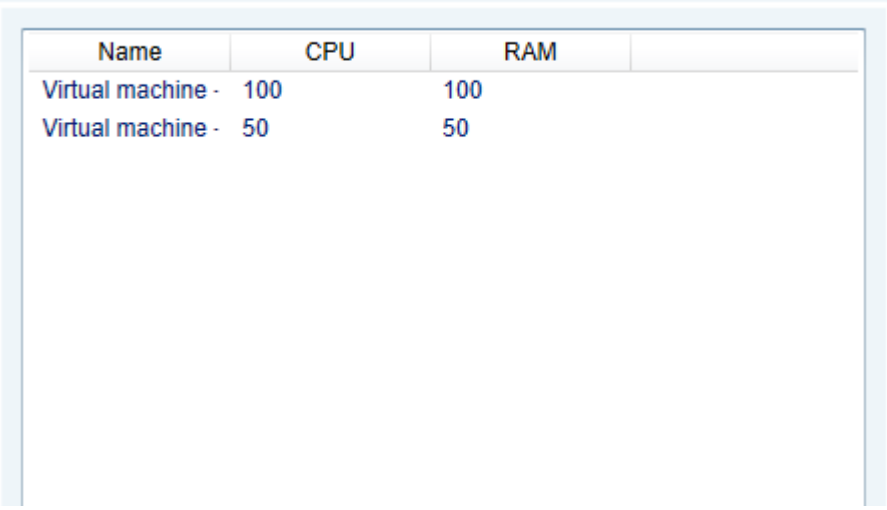


Рисунок 5.6 – Список віртуальних машин в сервері

Також, були розроблені модульні тести (Unit Testing), для можливості тестування методів розподілення ресурсів у високонавантажених системах.

5.4 Дослідницькі експериментальні результати

Для підтвердження ефективності використання методів розподілення ресурсів проведено ряд випробувань. Крім того, розроблені моделі експерименту, які дозволяють легко перевірити роботу алгоритму.

Під час експериментальних досліджень будемо використовувати віртуальні машини та сервери.

Для перевірки роботи методів розподілення ресурсів, було прийнято рішення порівнювати результати з послідовним розподіленням ресурсів та дисперсійним.

5.4.1 Дослідження дисперсійного методу

Оскільки сама по собі дисперсія має властивість виміряної мінливості. Таким чином, для вирішення задачі використане середнє значення та так звану волатильність. Волатильність представляє собою ризик, або так званий ризиковий показник, який можна використати при придбанні чогось або для досягнення якоїсь цілі.

При плануванні та при побудові дисперсії для набору чисел можна побачити, що дисперсія буває нулем тільки в одному випадку - коли всі числа які були в наборі чисел є нулями.

На рис. 5.7 зображено, як саме впливає кількість запитів на кількість віртуальних машин. За допомогою дисперсії можна чітко визначити середнє значення запитів за останні *n*-годин, проаналізувати та розподілити ресурси в центрі обробки даних. З рисунку можна бачити, що дисперсійне розподілення є ефективним.

Для того, щоб можна було чіткіше побачити різницю між послідовним розподіленням та дисперсійним, було створено тести, які відображені в таблиці 5.1, яка описує скільки було виконано запитів до сервера, та скільки саме часу було виділено на це. Ці тести виконували завдання навантаження сервера великою кількістю запитів, щоб можна було чітко зрозуміти та побачити, що саме відбувається з ресурсами під час великого навантаження.

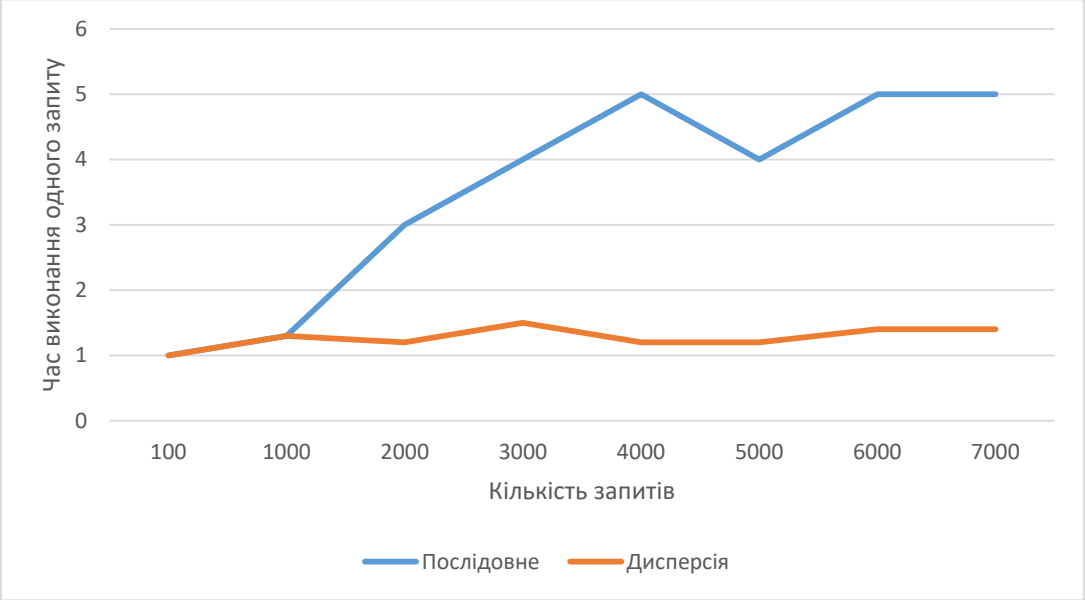


Рисунок 5.7 – Діаграма залежності віртуальних машин від кількості запитів до сервера

Таблиця 5.1 Опис навантаження сервера по кількості запитів та віртуальних машин

Кількість запитів	Послідовне розподілення ресурсів		Дисперсійний метод заповнення	
	Час виконання запиту	К – ть віртуальних машин	Час виконання запиту	К – ть віртуальних машин
100	1	1	1	1
1000	1.3	1	1.3	1
2000	3	1	1.2	2
3000	4	1	1.5	2
4000	5	2	1.2	4
5000	4	3	1.2	5
6000	5	4	1.4	7
7000	5	4	1.4	8

З таблиці 5.1 можна бачити, що навантаження було оброблено однаково для послідовного та дисперсійного розподілення лише при 1000 запитів до сервера. Далі, при збільшенні кількості запитів до 2000, час відповіді збільшився вдвічі. Таким, чином якість надання послуг наданих сервером, погіршились удвічі Що є неприйнятним для користувачів. Для покращення було використано дисперсійний метод, який проаналізував кількість та тривалість запитів, і зробив висновок, що кількість ресурсів потрібно збільшити.

Розподіл навантаження процесора в часі можна бачити на рис. 5.2. , процесор був завантажений на 90% і майже перевищив ліміт ресурсів сервера. Під час тестів, дисперсійний метод не переходив за 65% навантаження процесорів. Це пов’язано з тим, що при дисперсійному методі навантаження розподіляється між декількох віртуальних машин. Що дає нам можливість забезпечувати якісний контроль ресурсів в високонавантаженій IT-інфраструктурі.

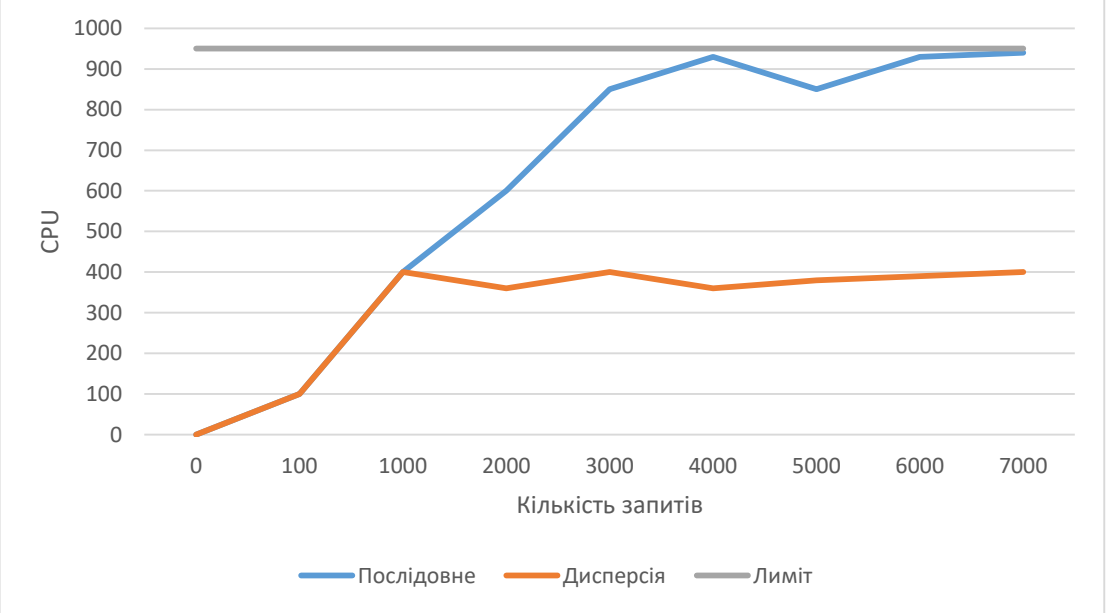


Рисунок 5.2 – Діаграма залежності запитів до процесора з дисперсією

5.4.2 Дослідження рекурентного методу

Для вирішення проблеми розподілу ресурсів необхідно здійснити передбачення, навантаження в системі протягом певного часу, які саме ресурси потрібні

для обслуговування користувачів та надання якісного сервісу. Одним із найцікавіших та швидких підходів представляється як зворотні співвідношення, які можуть аналізувати дані та якось керувати при потраплянні в ті характеристики, що є не підходящі для нас.

Зворотні співвідношення, як правило використовують для вирішенні зворотніх співвідношень. Рекурентні співвідношення часто зустрічаються в дискретній математиці і комбінаториці, тому сам підхід розробляє функції для вирішення зворотних співвідношень.

Таблиця 5.2 Опис навантаження сервера по кількості запитів та центрального процесора.

Кількість запитів	Послідовне розподілення ресурсів		Рекурентний метод	
	Центральний процесор	К – ть віртуальних машин	Центральний процесор	К – ть віртуальних машин
100	100	1	100	1
1000	400	1	400	1
2000	600	1	550	2
3000	850	1	500	2
4000	930	2	550	3
5000	850	3	480	4
6000	930	4	500	5
7000	940	4	510	6

Рекурентний метод буде тестуватись в таких же умовах, що і дисперсійний. Було створено велику кількість запитів до серверу та здійснено контроль операцій введення та виведення. Для рекурентного методу було досліджено також навантаження центрального процесору. Саме від нього, будемо відштовхуватись при розподіленні ресурсів з використанням рекурентного методу. Кількість запитів та завантаженість центрального процесора під час цих тестів, можна побачити в таблиці 5.2.

З таблиці 5.2 можна бачити, що завантаженість центрального процесора була однаковою до 1000 запитів. Після того, як кількість запитів збільшилась

вдвічі, навантаження на центральний процесор теж виросло. Тому рекурентний метод використав відсоток навантаження на процесор та зрозумів, що різниця між навантаження на 1000 та 2000 запитів, була значною і потрібно збільшити кількість виділених ресурсів для підтримання якості наданих послуг на прийнятному рівні.

Для візуального зображення навантаження в рекурентному підході був створений рис. 5.3 на якому можна бачити час пікового навантаження при запитах користувачів до системи. З рисунку 5.3 можна зробити висновок, що послідовне розподілення ресурсів не є доцільним, а рекурентний метод значно ефективніше розподіляє ресурси між користувачами та може бути використаний для забезпечення якості обслуговування клієнтам у високонавантажених системах.

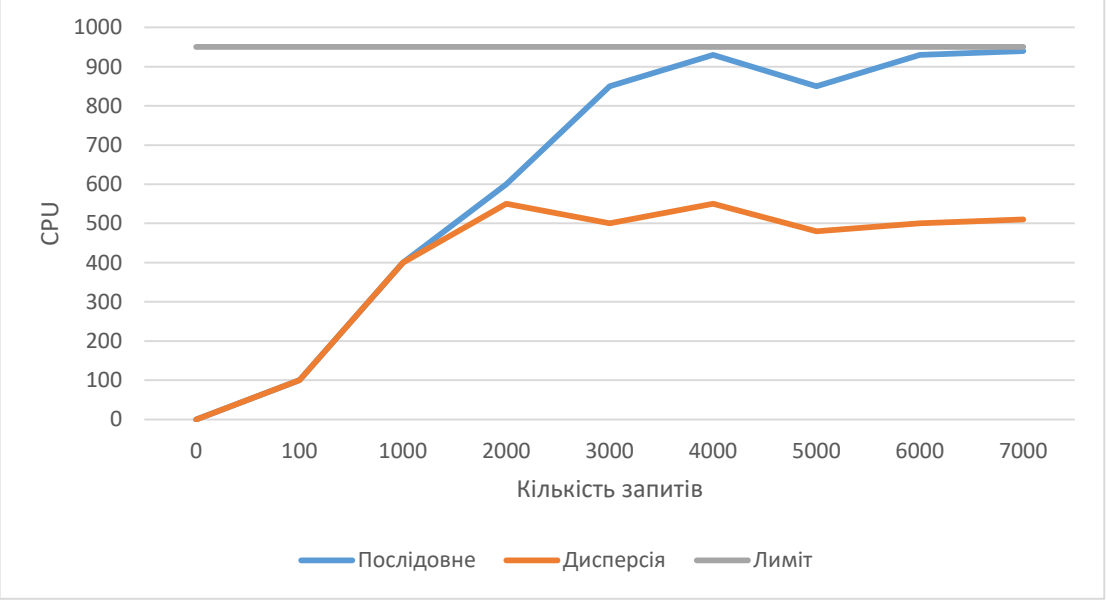


Рисунок 5.3 – Діаграма залежності запитів до процесора з рекурентним методом

ВИСНОВКИ ДО РОЗДІЛУ 5

В даному розділі було проведене тестування розробленого програмного забезпечення та методів розподілення ресурсів високонавантажених сервісів в IT-інфраструктурі.

Тестування показало, що при навантаженні системи, послідовне розподілення ресурсів було не ефективним – привело до погіршення якості наданих послуг. При розподілені ресурсів на основі дисперсії та рекурентного співвідношення були виконанні всі умови для забезпечення якісного надання послуг для користувачів.

Отже, використання послідовного розподілення ресурсів є недоцільним, а підхід до розподілення ресурсів на основі дисперсії та рекурентного співвідношення є ефективним та рекомендується використовувати у високонавантажених системах, оскільки він дозволяє забезпечити якість наданих послуг на високому рівні.

6 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Стартап проекти є популярними в усьому світі. Основною проблемою, стартап-проектів є те, що успіх ідеї не гарантується і він може сьогодні бути, а завтра вже ні. Ось чому, стартап проектів дуже багато, але тільки одиниці з них, можуть досягти успіху.

6.1 Опис ідеї

Основна ідея полягає в тому, щоб підтримувати якість послуг на узгодженому рівні. Система управління IT-інфраструктурою використовуючи методи масштабування ресурсів, їх розподілу та перерозподілу додає відповідним додаткам або ВМ, що підтримують високонавантажені сервіси, додаткові об’єми обчислювальних ресурсів.

Методи масштабування ресурсів для розподілу та перерозподілу:

Затримка трафіку: сюди входять затримка від введення в систему до бажаного результату. Затримка може бути розглянута через призму інфраструктури, протоколів та ПЗ.

Історія аналізу: допомагає знаходити можливі варіанти (дні, години, місяці) навантаження в системі для запобігання погіршенню якості послуг.

Експоненціальне зростання: зростання навантаження, коли швидкість росту пропорційна значенню самої величини. Якщо навантаження зростає протягом деякого періоду часу, то не чекаючи максимуму – виділяємо додаткові ресурси.

Кількість повідомлень: спостерігати за кількістю повідомлень для виконання певних дій.

Умовне зростання: сюди відносяться лінійна залежність – зростання та вихід за початкові умови.

Зміст ідеї разом з напрямками її застосування та перевагами(вигодами) для користувача наведено у таблиці 6.1.

Таблиця 6.1 опис ідеї стартап – проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Зміст ідеї є розроблення методів для управління ресурсами та якістю надання послуг для користувачів високонавантажених сервісів в IT-інфраструктурі з підвищенням ефективності використання ресурсів в центрах обробки даних.	Може застосовуватися як в цивільній так і в приватних сферах для підвищення якості наданих послуг.	За рахунок економії ресурсів, клієнти мають можливість економити кошти та свої ресурси, а також надавати кращі послуги для споживачів за такі самі кошти або менші. Для підприємців економія ресурсів та можливість надання кращих послуг для клієнтів.

Після пошуку можливих конкурентів, що займаються управлінням ресурсів та навантаженням. Було знайдено, декілька конкурентів, серед яких: AWS, Azure. Використовуючи звичайну систему управління віртуалізацією не дивлячись на те, що вона автономна та розумна, вона все одно потребує втручання людини. Говорячи, про якість управління, прямих конкурентів у відкритому доступі знайдено не було, це збільшує шанс, того, що проект може виявитись популярним та успішним в своїх сфері.

Інноваційність розробки полягає у використанні аналізу трафіку, експоненціального зростання та передбачення навантаження для запобігання погіршення умов для користувачів.

Конкуренти, що були знайдені можна віднести:

– конкурент 1 – система AWS.
– конкурент 2 – система Azure.

Таблиця 6.2 визначення сильних, слабких та нейтральних характеристик проекту

№	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона) а)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент1	Конкурент2			
1	Методи розподілу	опціональна	опціональна	опціональна	варіюється		присутня
2	Можливість передбачення навантаження	висока	відсутня	відсутня			присутня
	Вартість впровадження	середня	Висока	Висока	Складність реалізації		
	Технології розробки	Asp.net core	–	–			Новітні технології

6.2 Технологічний аудит ідеї

Розробка будь-якого додатку базується на проектуванні та виборі технологій, що можуть буди використанні при розробці даної системи. Потрібно знайти підходи та можливі варіанти вирішення поставлених завдань.

Аудит слугує для того, щоб здійснити аналіз підібраних технологій, які можуть бути використані для реалізації проекту. Результати технологічного аудиту наведено в таблиці 6.3.

Таблиця 6.3 Технологічний аудит проекту

№	Ідея проекту	Технології реалізації	Наявність технології	Доступність технології
1	Двигун для розробки з C#	Asp.net Core	+	+
2	Програмний каркас для побудови моделей	WPF	+	+
3	Мова програмування	C#	+	+
4	Мова програмування	Java	+	+
5	Мова програмування	Scala	+	+
6	Мова програмування	Python	+	+
7	Мова програмування	F#	+	+
Обрана технологія реалізації ідеї проекту: Asp.Net Core, WPF				

Запропонований варіант реалізації допоможе написати алгоритм та представити все графічному та зручному вигляді за допомогою WPF. Також, написаний алгоритм може бути використаний на різних операційних системах – це додає йому додаткову гнучкість.

Користувачу потрібно відобразити навантаження та розподіл ресурсів, для цього було використано технологію WPF. Розробка алгоритму була здійснена з використанням мови C# та ASP.NET Core використовуючи методологію DDD.

Наступним кроком буде визначення ринкових можливостей стартап-проекту, вони наведені в таблиці 6.4.

Таблиця 6.4 Характеристика потенційного ринку проекту

№	Показники стану ринку	Характеристика
1	Кількість конкурентів, од	2
2	Динаміка ринку	зростає
3	Запити або обмеження для потрапляння на ринок	немає
4	Вимоги до сертифікації чи стандартизації	присутні
5	Середня норма рентабельності на ринку, %	80%

Після аналізу потенційного ринку, куди буде вводитися проект, можна бачити, що кількість конкурентів не велика, але вони є передовими компаніями на ринку.

Так як, сфера розподілення ресурсів та IT-інфраструктури є непростою, попит у ній знаходиться на високому рівні, тому розробка даної системи та можливість її виводу на ринок є здійсненними.

Перед тим, як запроваджувати дану систему на ринок повинні бути чітко встановленні можливі варіанти виводу системи. Для того, щоб правильно вивести систему на ринок, має бути проведений аналіз конкурентів. У таблиці 6.5 був запропонований аналіз існуючих конкурентів. Для запровадження даної системи, можна використати декілька варіантів, а саме:

– запропонувати вже існуючим компаніям для покращення їх сервісу;
– надавати послуги, як додатковий сервіс.

В таблиці 6.5 розглянуто цільові групи клієнтів та основні вимоги споживачів, які будуть виокремлювати дану систему від всіх інших на ринку. Цільова група проектів є досить широкою, з її допомогою можна правильно керувати системою та запровадити її на ринок.

Основні фактори, що можуть впливати на продукт – фактори загроз та фактори можливостей.

Таблиця 6.5 Характеристика потенційних клієнтів стартап – проекту

Потреба ринку	Цільова аудиторія	Відмінність цільових груп клієнтів	Вимоги споживачів
Розподілення навантаження та забезпечення якості послуг	Приватні компанії	Система/технологія може використання для розподілу навантаження там можливості контролювати якість послуг. Може бути значним покращенням для компаній.	Висока швидкість забезпечення якості, Низький помилок при розподіленні навантаження, Можливість передбачити навантаження, як найшвидше, для якіснішого забезпечення послуг.
	Відкриті проекти	Метод, який може слугувати покращенням для інших компаній та економії ресурсів.	
	Люди, що хочуть покращити свої послуги	Можна зробити інтеграцію методів\ підходів для інших сервісів, таким чином, щоб люди могли цим користуватись не залежно де вони обслуговуються.	

Фактори загроз представляються собою загрози з якими можна зіштовхнутись при розробці системи, проектування або швидкості спрацювання системи. В такому випадку, вся суть проекту буде втрачена.

Фактори можливостей відображають проблему, яку система намагається вирішити – навантаження, розподілення, віртуалізація чи висока точність. Ризики описані в таблиці 6.6. Можливості в таблиці 6.7.

Таблиця 6.6 Фактори загроз

No п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Кваліфікація розробників	Недостатній рівень розробників в цій предметній області та пробіли в технічних знаннях.	Можливий варіант пошуку нових кадрів на ринку для можливості реалізації функцій та програмного забезпечення. Також, можливий варіант використовувати соціальні мережи для розробників або спеціальні сайти. Для того, що ім'я компанії було популярним можна використовувати конференції.
2	Конфеденційні матеріали	Порушення цих прав та розголошення інформації, що стосується тільки компанії.	Використовуючи юристів потрібно скласти договір, який буде забезпечувати конфіденційність інформації за законами України. За порушення цих правил особа нестиме кримінальну відповідальність.
3	Швидкодія роботи	Відносно інших систем швидкість може бути замала	Використовувати можливі тести для заміру швидкості та знаходити слабкі місця в системі.
4	Поява конкурентів	Можлива поява конкурентів	Розвинення та впровадження нових можливостей.

Фактори загрози, що можуть бути під час введення компанії на ринок було наведено в таблиці 6.6. Після того, як всі загрози були нівельовані, можна звернути увагу на фактори можливостей, що наведені в таблиці 6.7.

Таблиця 6.7 Фактори можливостей

No п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Затребуваність продукту	Чим більше буде затребуваність до продукту, тим більше буде клієнтів та проект буде розвиватись	Впровадження нового функціоналу в систему
2	Висока точність передбачення та розділення ресурсів	Можливість знаходження нових клієнтів для розширення області використання додатку	Залучення нових інвесторів для збільшення продукту
3	Відкритий проект	Можливість реклами та покращення продукту	Отримання нового функціоналу та популярності проекту без коштів

Таблиця 6.8 Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вид конкуренції – ринкова конкуренція	Конкуренція відбувається завдяки якості та кількості функціональності в продукті	Покращення швидкості та зручності користування продуктом, також може бути присутня реклама та робота з користувача про отримання відгуків

Після знаходження факторів можливостей, можна приступити до аналізу конкуренції у сфері надання послуг розподілення та надання якісних послуг, результати аналізу наведено у таблиці 6.8.

Продовження таблиці 6.8

Конкурентні середовища	Проявлення даної характеристики	Конкурентоспроможність компаній на ринку
2. Конкурентна боротьба	Використання продукту в різних країнах	Ознайомитись із законами країни, де буде використовуватись продукт і що саме необхідно для кожної з країн.
3. Галузь у внутрішньогалузевій	Продукт використовується в хмарних вебсервісах	Базовий функціонал, який буде доступний для всіх країн. І можливість розширення функціоналу.
4. Можливість конкуренції	Існують деякі методи для забезпечення якісного використання ресурсів	Реалізація базового функціоналу. Отримання максимуму швидкодії та точності.
5. Конкурентні переваги та їх характеристики	Головним фактором є якість розподілення ресурсів та забезпечення якісної роботи, швидкість та стабільність роботи методів розподілення.	Запровадження та покращення методів для вирішення проблем з якісним забезпеченням послуг
6. Особливості конкурентного середовища	Проявлення конкретності	Бути конкурентоспроможною компанією з розробленням нових методів та підходів

Перед тим, як вводити компанії та системи на ринок, потрібно мати уявлення про конкурентів тому треба здійснити аналіз конкуренції. Можна використати метод п’яти сил конкуренції Майкла Портера. Результати аналізу наведені у таблиці 6.9

Таблиця 6.9 Аналіз за методом п’яти сил конкуренції Майкла Портера

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари – замінники
Складові аналізу	Відсутні	Присутні	Відсутні	Проект орієнтований хмарні веб-сервіси, тому вони матимуть можливість покращувати методи для покращення якості послуг	В наявності Система AWS, Kubernetes
Висновки:	Конкурентів з такими методами на даний момент немає, але є з подібними	Незважаючи на всіх конкурентів можливий варіант виходу на ринок	Постачальники не мають цих можливостей, які реалізують наші методи	Основною можливістю якісного контролю ресурсів та навантаження.	Обмежень ніяких немає

Підсумовуючи вище написаний аналіз про систему, можна зробити висновок що старт проекту можливий і він буде успішним.

Таблиця 6.10 Обґрунтування факторів конкурентоспроможності

No п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Можливість аналіз трафіку	Досягається за допомогою моніторингу часу, який потрібний на оброблення запиту
2	Можливість аналізу графіків навантаження	Досягається за рахунок використання історії навантаження
3	Можливість обробки повідомлень, що чекають своєї черги	Досягається шляхом отримання кількості повідомлення, та визначення навантаження на сервіс
4	Можливість врахування експоненціального зростання	Досягається за рахунок використання аналізу по зростанню запитів

Таблиця 6.11 Порівняння сильних та слабких сторін проекту

No п/п	Фактор конкурентоспроможності	Бали 1 – 20	Рейтинг товарів – конкурентів						
			– 3	– 2	– 1	0	+1	+2	+3
1	Можливість аналізу трафіку	18			K1	K2			+
2	Можливість аналізу графіків навантаження	19		K1	K2				+
3	Можливість обробки повідомлень, що чекають своєї черги	20		K1	K2				+
4	Можливість врахування експоненціальне зростання	20	K1	K2					+

На основі отриманих даних можна провести обґрунтування властивостей, які виділяють проект серед конкурентів і забезпечують конкурентоздатність на ринку. Його результати наведено у таблиці 6.10. На основі визначених методів та їх особливостей було здійснено аналіз методу з іншими. Результати порівняння наведені у таблиці 6.11.

Результати де було вказано сильні та слабкі сторони, загрози та всі можливості системи (SWOT) наведені у таблиці 6.12.

SWOT представляє собою метод для стратегічного планування, що полягає в виявленні факторів для внутрішнього та зовнішнього середовища, і поділяю їх на чотири категорії.

Таблиця 6.12 SWOT – аналіз стартап-проекту

Сильні сторони: Використання аналізу трафіку, графікі та інформації технології машинного навчання; Надає розробці функціонал, який відсутній у конкурентів;	**Слабкі сторони:** Складне розгортання Важка розробка
Можливості: Збільшення кількості ресурсів та контроль якість навантаження шляхом надання функціоналу якому немає аналогів на даний момент	**Загрози:** Недостатня експертиза розробників Нові конкуренти з аналогічним функціоналом

В таблиці 6.12 були описані сильні сторони та можливості для стартап – проекту.

Після проведення SWOT аналізу потрібно розробити альтернативні реалізації від часу, підібраних технологій. Можливі варіанти реалізації наведені в таблиці 6.13.

Для розроблення системи були обрано ASP.NET Core для написання алгоритму, який буде аналізувати трафік при затримці, експоненціальне зростання та робити аналіз даних з завантаженням.

Конкуренція можлива, але на даний момент її немає, так як жодна система немає таких методів для розподілення навантаження та аналізу даних, але треба враховувати також час для розроблення даної системи та запровадження її га ринку. Можливі варіанти реалізації наведені в таблиці 6.13.

Таблиця 6.13 Альтернативи ринкового впровадження стартап-проекту

No п/п	Альтернатива ринкової поведінки	Імовірність балансування ресурсів та забезпечення якості сервіса	Термін реалізації
1	Розробка системи з використанням ASP.NET Core, аналізу про затримку трафіка, експоненціальне зростання та аналізу даних з завантаженням	Висока	24-30 місяців
2	Розробка системи без використання аналізу даних	Середня	8-10 місяців
3	Розробка системи без аналізу експоненціального зростання	Низька	12-18 місяців
4	Розробка системи без аналізу затримки трафіка	Низька	18-24 місяців
5	Розробка системи з використанням ASP.NET Core	Середня	12-18 місяців

В таблиці 6.13 альтернативи ринкового впровадження стартап – проекту було описано можливий час для розробки даної системи в залежності від того, що повинно бути в системі, і що на чому саме система буде основуватись.

6.3 Розроблення ринкової стратегії проекту

Для створення успішного продукту, потрібно знати цільову групу для системи.

Таблиця 6.14 Цільова групи споживачів

№	Цільова група потенційних клієнтів	Готовність споживачів отримати продукт	Попит меж в цільових групи (сегменту)	Конкуренція сегментів	Простота виходу на ринок
1	Приватні компанії	Є потреба та мають готовність прийняти продукт	Сильний попит в сильно – навантажених системах	Конкуренція можлива, але на даний момент немає	Запровадження продукту не буде представляти проблем
2	Відкриті проекти	Є потреба та мають готовність прийняти продукт	Сильний попит в сильно – навантажених системах	Конкуренція можлива, але на даний момент немає	Запровадження продукту не буде представляти проблем

Також, розуміти попит груп компаній чи людей, та звичайно ж готовність споживачів отримати продукт та користуватись ним у себе к компанії або проекті. Для цього було чітко визначена цільова група споживачів наведено, що була наведена у таблиці 6.14.

На таблицях були зображені цільові групи. Не дивлячись на те, яка з груп буде вибрана треба налагодити зв’язки з її представникам для того, щоб запропонувати реалізацію продукту. Але в цьому можуть виникнути певні труднощі.

Продовження Таблиці 6.14

№	Цільова група потенційних клієнтів	Готовність споживачів отримати продукт	Попит меж у цільових групи (сегменту)	Конкуренція сегментів	Простота виходу на ринок
3	Компанії, що хочуть покращити свої послуги	Є потреба та мають готовність прийняти продукт	Сильний попит в сильно – навантажених системах	Конкуренція можлива, але на даний момент немає	Запровадження продукту не буду представляти проблем
Які цільові групи обрано: приватні і відкриті проекти, а також люди\ компанії, що хочуть забезпечити якісне керування ресурсами.					

Наступний крок – визначення стратегії розвитку продукту, таблиця 6.15.

Таблиця 6.15 Визначення базової стратегії розвитку

Альтернатива розвитку проекту	Стратегія отримання ринку	Ключові конкурентос-проможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розробка системи з використанням ASP.NET Core, аналізу про затримку трафіка, експоненціальне зростання	Концентрації	Конкуренція у сфері хмарних веб-сервісів, які мають можливість реалізувати такий самий функціонал	Диференціації

Після визначення базової стратегії потрібно визначити стратегію поведінки для конкурентів. Результати наведено у таблиці 6.16.

Таблиця 6.16 Визначення стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Можливість компанії шукати нових споживачів або хотіти забрати існуючих конкурентів?	Чи буде мати можливість компанія реалізовувати методи продукту конкурента?	Стратегія конкурентної поведінки*
Ні	Так	Жодних	Експлерентна (піонерська)

Після обраних сегментів споживачів, та потреб для обраної стратегії поведінки потрібно вибрати стратегі. Для позиціонування продукту. Результати наведемо у таблиці 6.17.

Таблиця 6.17 Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні варіанти стартап-проекту	Комплексна позиція власного проекту
Немає	Диференціації	Точність забезпечення якісних послуг високонавантажених сервісів	Вчасне балансування ресурсів, та підняття додаткових ресурсів

6.4 Маркетингова програма в стартап-проекті

В даному розділі запропоновано маркетинговий проект для проекту. Описано недоліки та переваги, які може тримати користувач якщо буде використовувати систему. В таблиці 6.18 наведено основні переваги даного підходу

Таблиця 6.18 Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода яку пропонує функція	Ключові переваги перед конкурентами
1	Точність розпізнавання затримки трафіку	Можливість покращити якість надання послуг для користувачів за допомогою збільшення ресурсів	Не реалізовані, можна запровадити даний метод

Продовження Таблиці 6.18

№	Потреба	Вигода яку пропонує функція	Ключові переваги перед конкурентами
2	Експоненціальний зріст трафіку	Можливість системи передбачувати навантаження в якісь особливі дні або час	Не реалізовані, можна запровадити даний метод
3	Кількість запитів для обробки	Якщо система запрошує багато запитів, то потрібно визначитись з параметрами для якісного надання послуг	Не реалізовані, можна запровадити даний метод

В таблиці 6.18 було описано ключові переваги концепції потенційного продукту. Його плюси та мінуси, потреби та вигоди. Описано три потреби, які є обов’язкові для розробки системи, в інакшому випадку система не буду конкурувати з іншими розробками.

Для кожної потреби була описана можливість, яку отримає користувач при використанні системи. Після того, як описали вигоду також було коротко описано як конкурувати з конкурентами, що є в них, і що є в нашому продукті. Розглядаючи всі потреби, що були описані для прямих конкурентів з такими ж перевагами на даний момент немає. Розроблена система буде надавати ряд переваг:

– точність розпізнання затримки трафіка при опрацюванні запитів від користувача;

– експоненціальне навантаження, що дасть можливість системі передбачити масове навантаження;

– кількість запитів до системи протягом певного часу.

Таблиця 6.19. Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Система за задумом</td><td colspan="3">Підходи, які будуть аналізувати вхідні дані</td></tr>
<tr><td rowspan="6">II. Система у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Висока швидкість аналізу</td><td>М</td><td>Тх/Тл</td></tr>
<tr><td>2. Висока точність обробки з аналізом</td><td>М</td><td>Тх/Тл</td></tr>
<tr><td>3. Низький рівень фіктивних спрацювань</td><td>М</td><td>Тх/Тл</td></tr>
<tr><td colspan="3">Програма пройшла тестування, ISO/IEC 9126 – 1 2013</td></tr>
<tr><td colspan="3">Постачається в форматі алгоритму або скриптів</td></tr>
<tr><td colspan="3">Марка: ahumen, gen&humen, scale_group</td></tr>
<tr><td colspan="2" rowspan="2">III. Товар із підкріпленням</td><td colspan="2">До продажу Програмне забезпечення</td></tr>
<tr><td colspan="2">Після продажу Програмне забезпечення</td></tr>
<tr><td colspan="4">Патент методу обробки та аналізу даних для забезпечення якісних умов для користувача, законодавство про авторське право</td></tr>
</table>

Залишилось визначитись з межами ціни для продукту. Результати наведено у таблиці 6.20

Таблиця 6.20. Визначення меж встановлення ціни

Рівень цін (розробка)	Рівень цін на товари – аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на то – вар/послугу
100 тис гривень	Відсутні	Середній клас населення	Від 200 тис. Гривень за ліцензію

Було описано ціни, після чого можна розглянути варіанти збуту. Результати наведені у таблиці 6.21.

Таблиця 6.21. Формування системи збуту

No п/п	Специфіка поведінки цільових клієнтів	Можливості збуту, які виконує постачальник Функції, які повинен виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Використовуючи методи та підхід за допомогою скрипту або програмного забезпечення	Купівля через сайт	Від виробника до споживача	Через сайт компанії виробника

Останнє, що залишилось-концепція маркетингових комунікацій. Результати наведені в таблиці 6.22.

Таблиця 6.22 Концепція маркетингових комунікацій

Специфіка поведінки клієнтів	Канали якими користуються цільові клієнти	Ключові позиції	Завдання реклами	Концепція реклами
Оцінка вибору кращого продукту та отримання кращого сервіса	Сайтами компанії, компаніями, їхніми товарами і т.д.	Забезпечення якісного сервісу та контролю над ресурсами	Показати економію та можливість покращення вашого сервіса	Знаходження та контроль бізнесів за допомогою контролю ресурсів

ВИСНОВКИ ДО РОЗДІЛУ 6

В даному розділі було розглянуто розробку стартап-проекту системи для контролю якості надання послуг для користувачів за допомогою аналізу навантажень та розподіленню навантаження. Було наведено опис класів аномалій, проведений аналіз ринку та обрана цільову аудиторію, яка може отримати вигоду після використання даного продукту.

Так як прямих аналогів не було знайдено, єдиним фактором, що може затримувати розробку – інтеграція і можливість інтеграцій вже існуючі системи.

Фактори, які можуть впливати на продаж – ціна та використання або розгортання. Першим фактор є можливість коригування в залежності від попиту та бажань. Другий потребує аналізу та може бути покращений лише при здійсненні аналізу системи. Підсумовуючи вище написане, можна сказати, що проект є перспективним і подальша розробка його є доцільною та затребуваною.

ВИСНОВОК

У ході виконання магістерської дисертації була вирішена задача розробки та аналізу методів управління основаних на рекурентному та дисперсійному алгоритмів, з метою розподілення та керування ресурсів в центрах обробки даних.

Для аналітичного обґрунтування роботи системи були створені математичні моделі управління ресурсами та навантаженням. Також, було розроблено ряд обмежень на основі потужностей серверів в ЦОД. Для вирішення поставлених задач у роботі було розроблено два варіанти контролю розподілення ресурсів: рекурентний та дисперсійний. Разом ці два методи можуть розподіляти ресурси та керувати якістю надання сервісу для користувачів.

Результатом дослідження є розроблена модель засобу моделювання управління ресурсами та навантаженням, з метою покращення управління навантаженням. Було отримано гнучке рішення, яке дозволяє змінювати роботу алгоритму в залежності від задачі, що була поставлена перед ним.

Методи алгоритму були реалізовані у програмному забезпеченні, яке представляє собою інтерфейс для роботи з алгоритмом. Програмний продукт представляє собою бібліотеку класів, що була розроблена за допомогою ASP.NET Core з використанням DDD, і мови програмування C#.

Після проведеного аналізу та тестування методів для розподілу ресурсів, можна бачити, що алгоритм спроможний вирішити поставлену задачу та досягти ефективного використання ресурсів і гарантувати якісний сервіс для користувачів, навіть у високонавантажених сервісах.

Завдяки цим дослідженням та розробці маємо можливість застосовувати їх у реальних системах управління ресурсами та навантаженням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Buyya R., Calheiros R. N., Son J., Dastjerdi A. V., Yoon Y. Software – defined cloud computing: Architectural elements and open challenges/ Advances in Computing, Communications and Informatics (ICACCI) International Conference. – 2014. – pp. 1 – 12.

2. Jararweh Y., Al – Ayyoub M., Benkhelifa E., Vouk M., Rindos A. Software defined cloud: Survey, system and evaluation/ Future Generation Computer Systems. – 2015. – vol. 58, May 2016. – pp. 56 – 74. 3.

3. Hariri S., Khargharia B., Chen H., Yang J., Zhang Y., Parashar M., Liu H. The autonomic computing paradigm/ Cluster Comput. – 2006. – vol. 9 (1). – pp. 5 – 17.

4. Hameed A., Khoshkbarforoushha A., Ranjan R., Jayaraman P. P., Kolodziej J., Balaji P., Zeadally S., Malluhi Q. M., Tziritas N., Vishnu A. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems/ Computing. – 2014. – pp. 1 – 24.

5. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions/Future Generation Computer Systems. – 2013. – vol. 29, no. 7. – pp. 1645 – 1660.

6. S. Telenyk., O. Rolik, P.S. Savchenko and M. E. Bodaniuk, “Manageable genetic algorithm in tasks of distribution of virtual machines in data centres,” Visnyk of Cherkasy State Technological University), 2011. vol. 2, pp. 104 – 113,

7. Clark C., Fraser K., Hand S., Hansen J. G., Jul E., Limpach C., Pratt I., Warfield A. Live migration of virtual machines/ Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation. – 2005. – vol. 2. – pp. 273 – 286.

8. Wood T., Shenoy P. J., Venkataramani A., Yousif M. S. Black – box and Gray – box Strategies for Virtual Ma – chine Migration/ NSDI. – 2007. – vol. 7. – pp. 17 – 17.

9. Bobroff N., Kochut A., Beaty K. Dynamic placement of virtual machines for managing SLA violations/ Integrated Network Management, 2007. IM’07. 10th IFIP/IEEE International Symposium. – 2007. – pp. 119 – 128.

10. Zhu X., Young D., Watson B. J., Wang Z., Rolia J., Singhal S., McKee B., Hyser C., Gmach D., Gardner R. 1000 islands: Integrated capacity and workload management for the next generation data center/ Autonomic Computing (ICAC'08). International Conference. – 2008. – pp. 172 – 181.

11. Calder B., Wang J., Ogus A., Nilakantan N., Skjolsvold A., McKelvie S., Xu Y., Srivastav S., Wu J., Simitci H. Windows Azure Storage: a highly available cloud storage service with strong consistency. In Proceedings of the Twenty – Third ACM Symposium on Operating Systems Principles. ACM. – 2011. – pp. 143 – 157.

12. Shen Z., Subbiah S., Gu X., Wilkes J. Cloudscale: elastic resource scaling for multi – tenant cloud systems/ Pro – ceedings of the 2nd ACM Symposium on Cloud Computing. – 2011. – p. 5.

13. Beloglazov A. Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and perfor – mance efficient dynamic consolidation of virtual ma – chines in Cloud data centers/ Concurrency and Computa – tion: Practice and Experience. – 2012. – vol. 24, no. 13. – pp. 1397 – 1420.

14. Al – Ayyoub M., Jararweh Y., Daraghmeh M., Althebyan Q. Multi – agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure/ Cluster Computing. – 2015. – vol. 18, no. 2. – pp. 919 – 932.

15. Ролик А.И. Декомпозиционно – компенсационный подход к управлению уровнем услуг в корпоративных ИТ – инфраструктурах / А.И. Ролик // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+, – 2013. – № 58. – С. 78 – 88.

16. Ролик А.И. Управление уровнем услуг корпоративной ИТ – инфраструктуры на основе координатора / А.И. Ролик //Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. наук. пр. – К.: Век+. – 2013. – № 59. – С. 98 – 105.

17. Теленик С.Ф. Адаптивный генетический алгоритм для решения класса задач распределения ресурсов ЦОД / С.Ф. Теленик, А.И. Ролик, П.С. Савченко // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2011. – № 54. – С. 164 – 174.

18. Ролик А.И. Концепция управления корпоративной ИТ – инфраструктурой / А.И. Ролик // Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка. – К.: «ВЕК+», 2012. – № 56. – С. 31 – 55.

19.Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов // Автоматика. Автоматизація. Електротехнічні комплексі та системі. – 2010. – №1 (25). – С. 106 – 120.

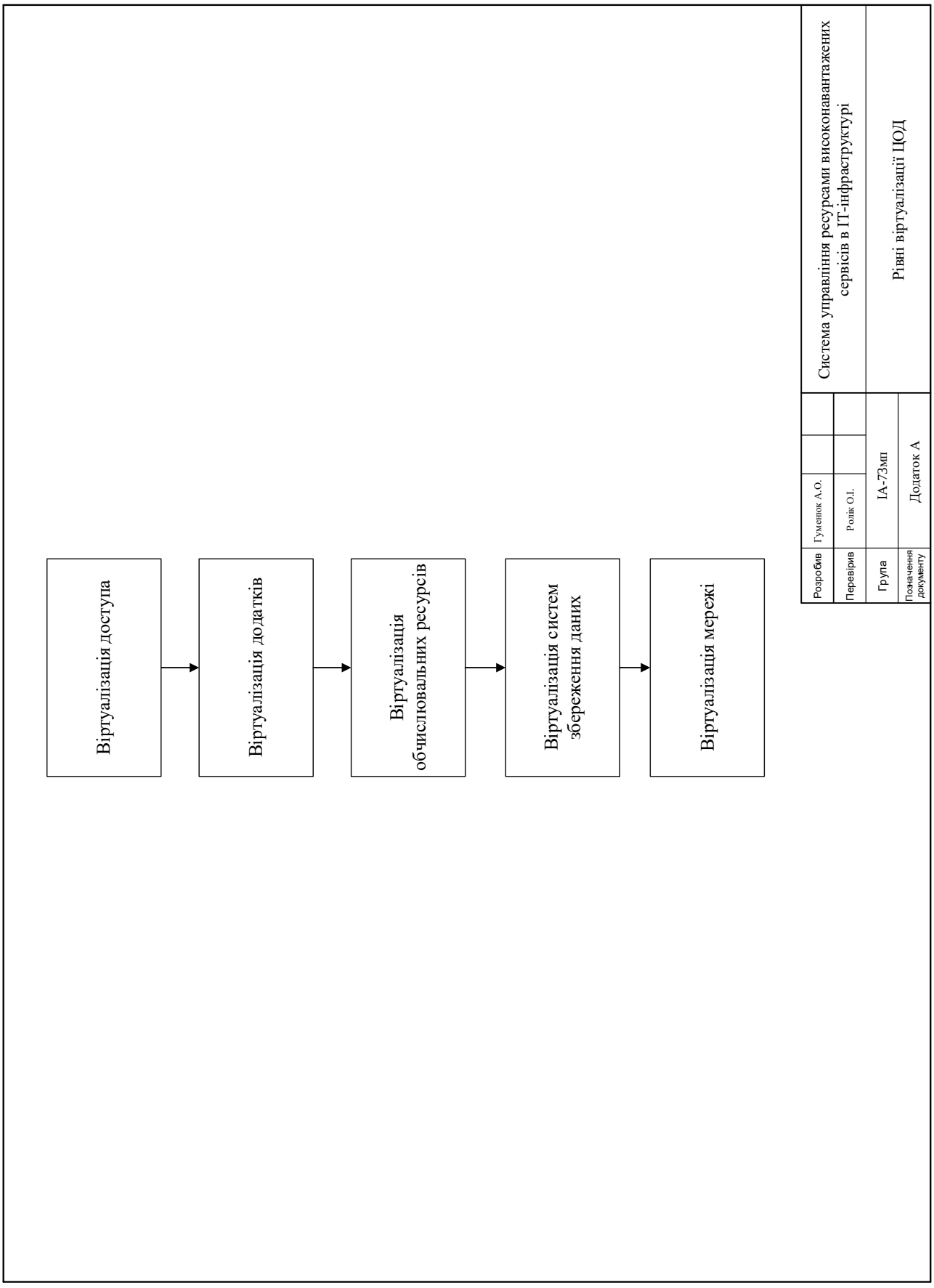
20. X. Dutreilh,A. Moreau, J. Malenfant, N. Rivierre, and I. Truck, “From data center resource allocation to control theory and back,”Cloud Computing, pp.410 – 417, 2010.

21. Calder B., Wang J., Ogus A., Nilakantan N., Skjolsvold A., McKelvie S., Xu Y., Srivastav S., Wu J., Simitci H. Windows Azure Storage: a highly available cloud storage service with strong consistency. In Proceedings of the Twenty – Third ACM Symposium on Operating Systems Principles. ACM. – 2011. – pp. 143 – 157.

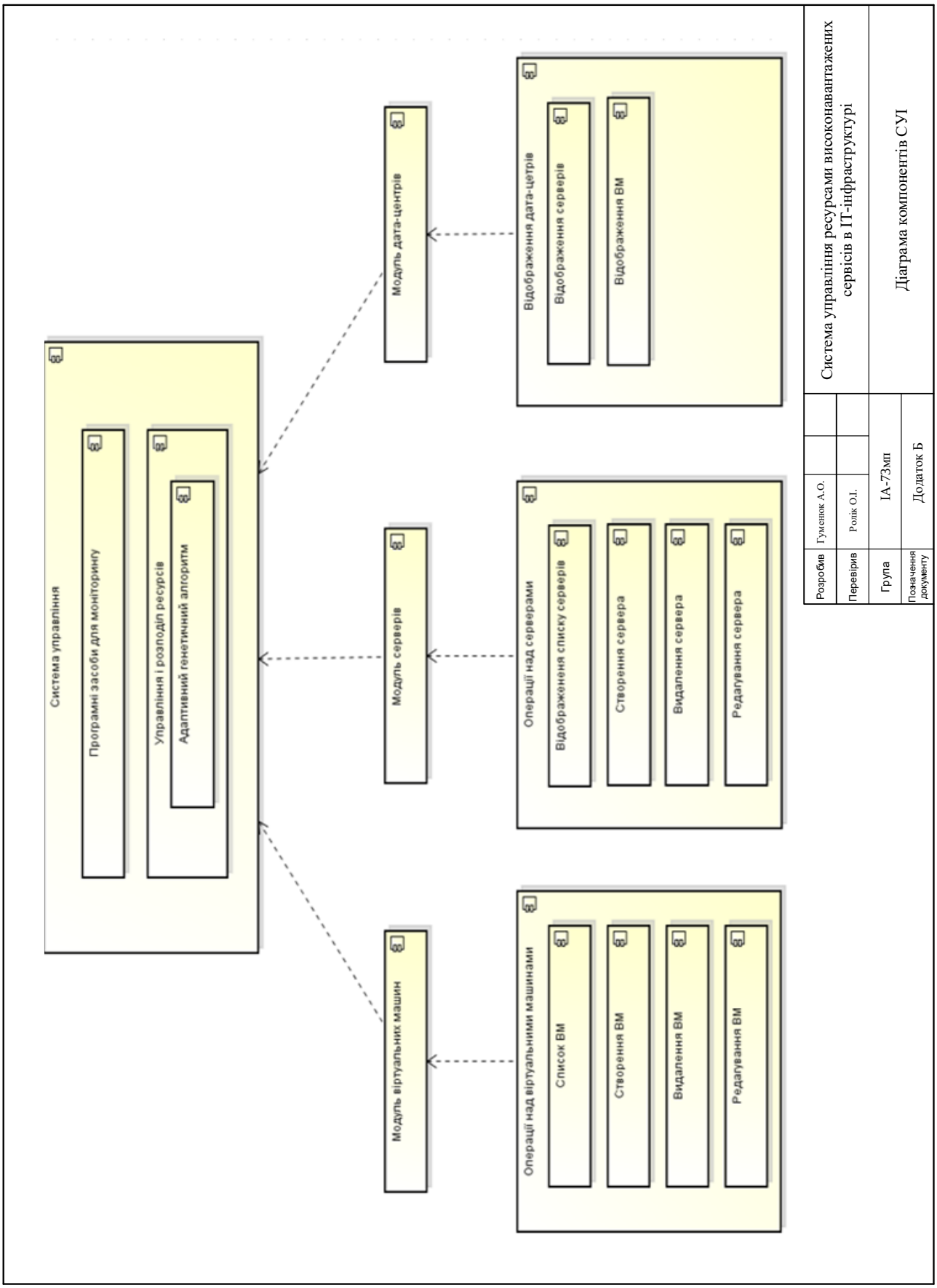
22. Rolik A. I. “Decomposition – compensation method of service level management of corporate IT infrastructures,” Visnyk NTUU “KPI” Informatics, operation and computer science, 2013, vol. 58, pp. 78 – 88.

23. Telenyk, S., Zharikov E., Rolik O., "Architecture and conceptual bases of cloud IT infrastructure management." Advances in Intelligent Systems and Computing. Springer, Cham, 2017,pp.41 – 62.

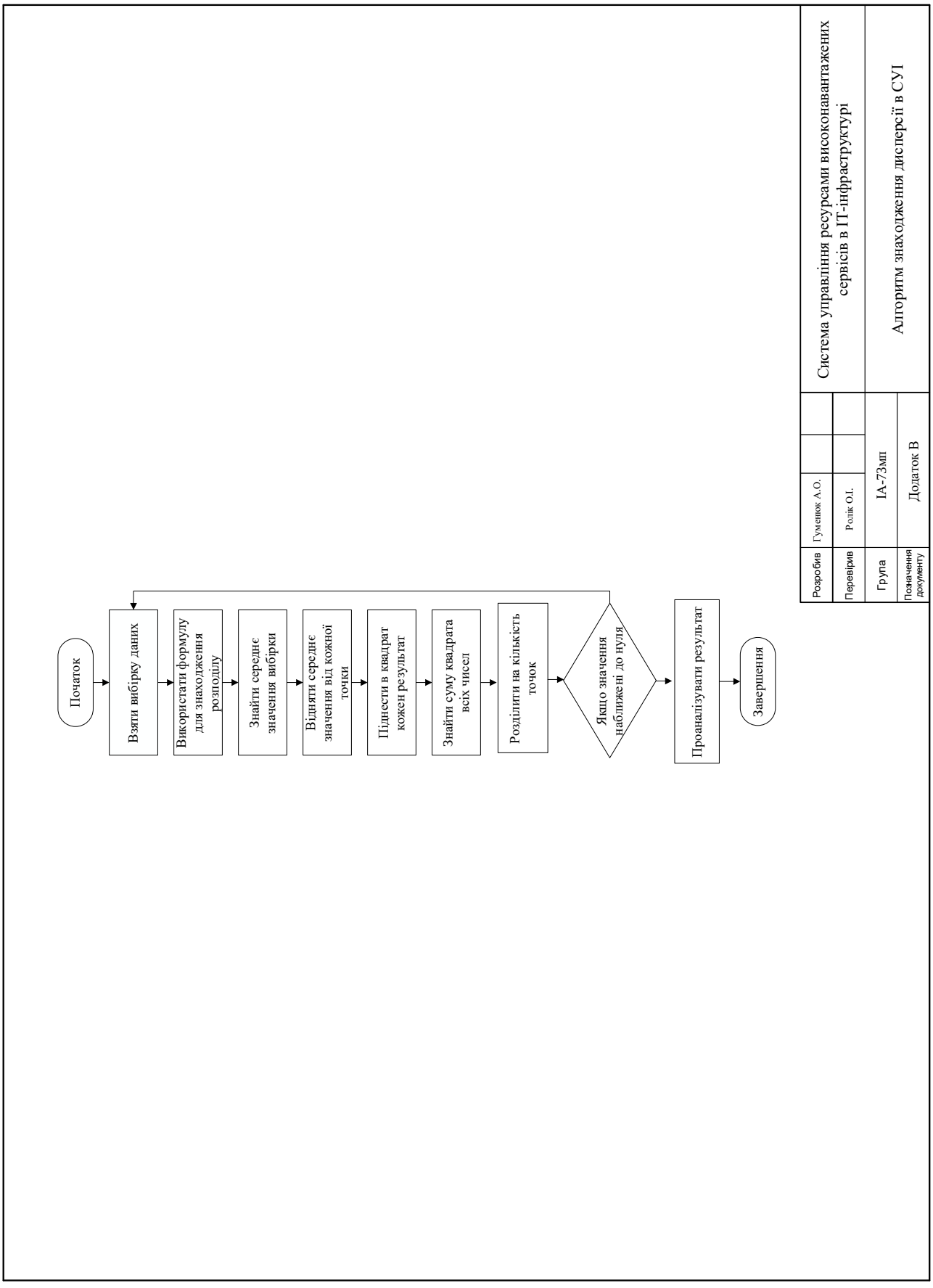
ДОДАТОК А – РІВНІ ВІРТУАЛІЗАЦІЇ ЦОД



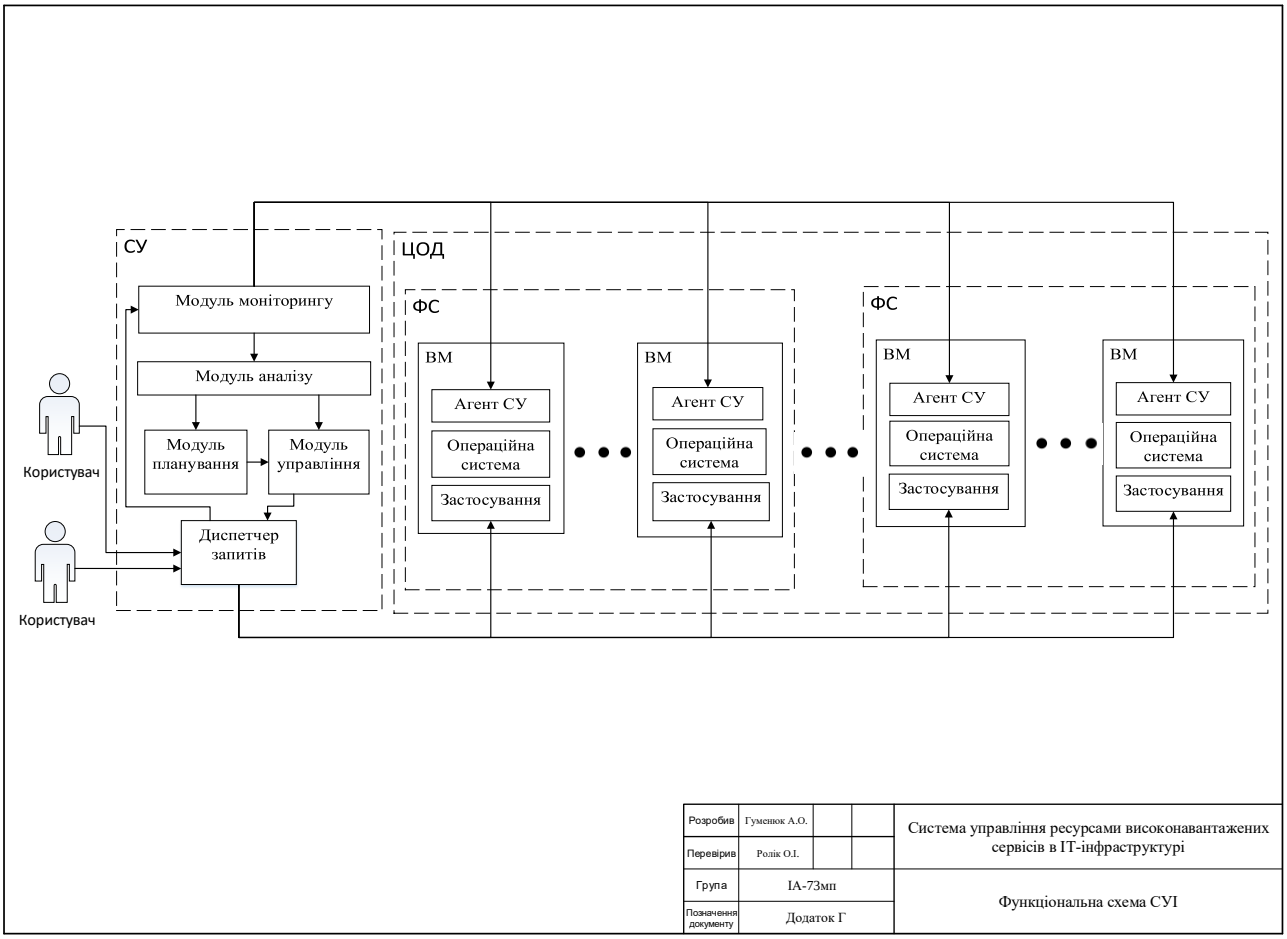
ДОДАТОК Б – ДІАГРАМА КОМПОНЕНТВ СУІ



ДОДАТОК В – АЛГОРИТМ ЗНАХОДЖЕННЯ ДИСПЕРСІЇ В СУІ

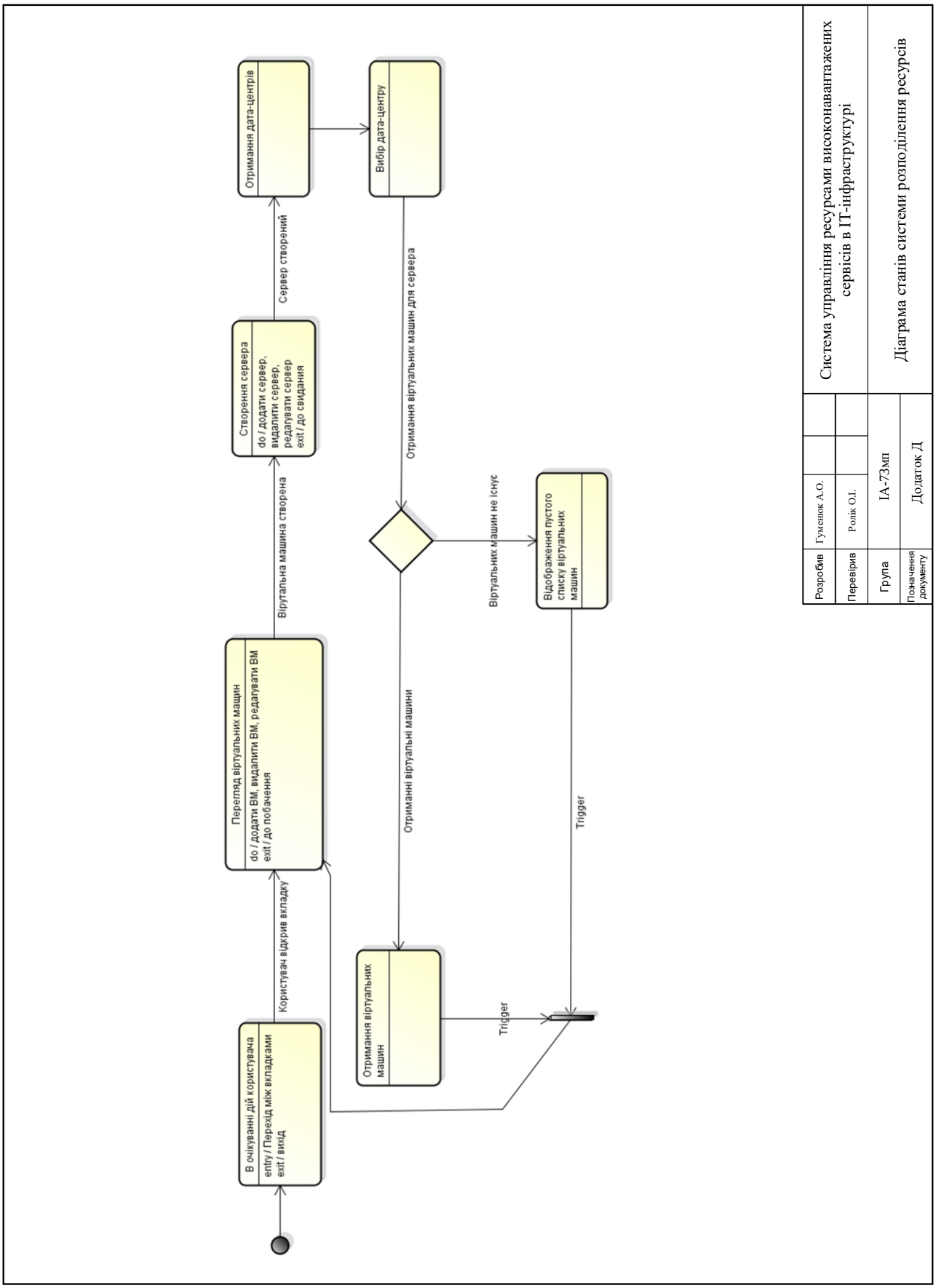


ДОДАТОК Г – ФУНКЦІОНАЛЬНА СХЕМА СУІ

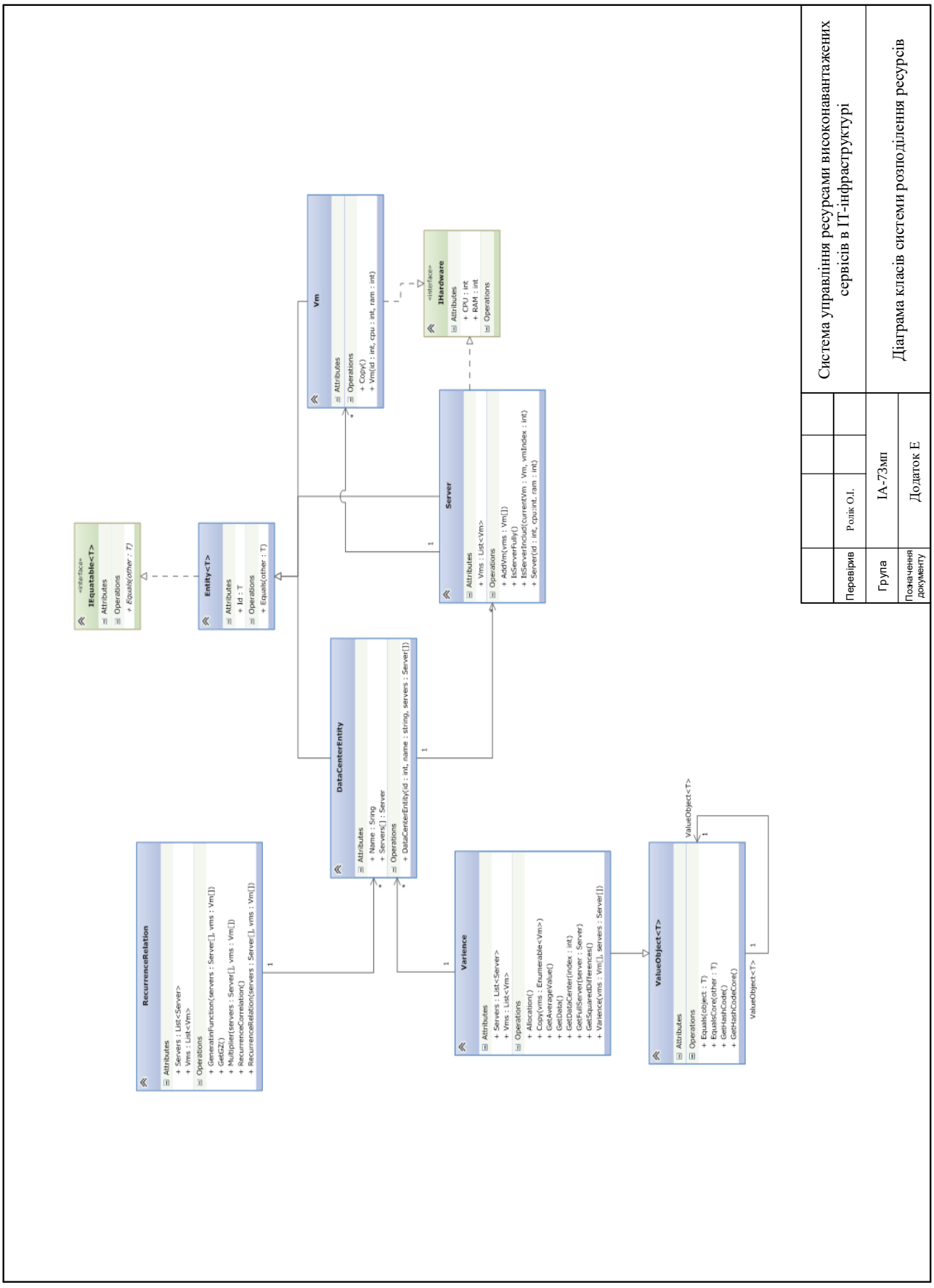


Розробив	Гуменюк А.О.			Система управління ресурсами високонавантажених сервісів в ІТ-інфраструктурі
Перевірив	Ролік О.І.			
Група	ІА-73мп			Функціональна схема СУІ
Позначення документу	Додаток Г			

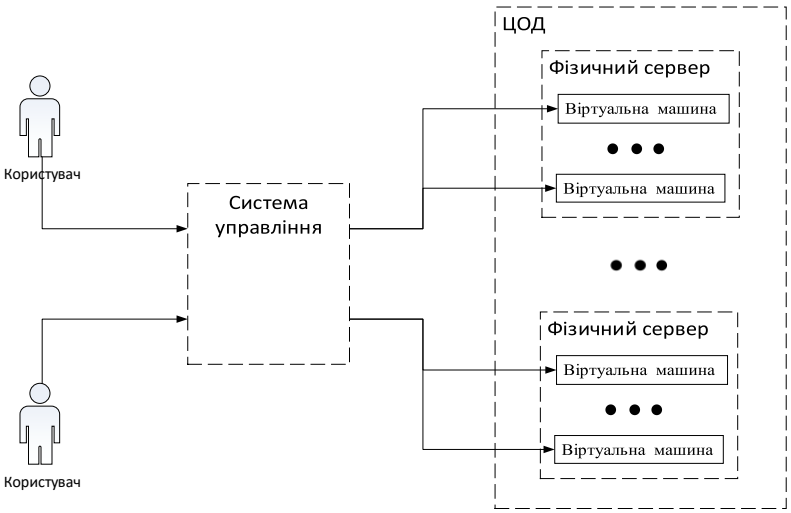
ДОДАТОК Д – ФУНКЦІОНАЛЬНА СХЕМА СУІ



ДОДАТОК Е – ДІАГРАМА КЛАСІВ СИСТЕМИ РОЗПОДІЛЕННЯ РЕСУРСІВ

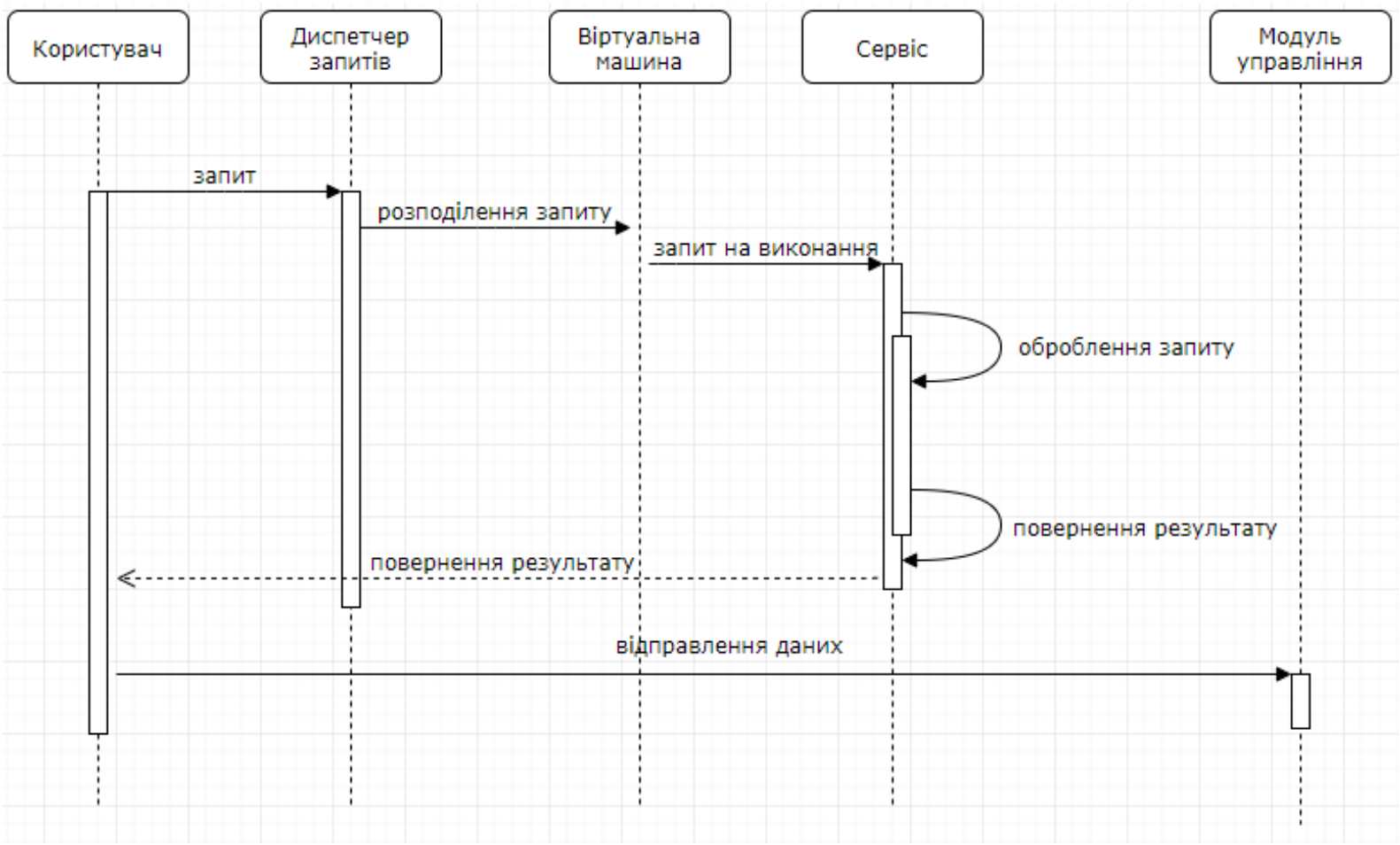


ДОДАТОК Є – СТРУКТУРНА СХЕМА СИСТЕМИ РОЗПОДІЛЕННЯ РЕСУРСІВ



				Система управління ресурсами високонавантажених сервісів в IT-інфраструктурі
Перевірив	Ролік О.І.			
Група	ІА-73мп			Структурна схема СУІ
Позначення документу	Додаток Є			

ДОДАТОК Ж – ДІАГРАМА ПОСЛІДОВНОСТІ ВЗАЄМОДІЇ КОРИСТУВАЧА З СУІ



				Система управління ресурсами високонавантажених сервісів в IT-інфраструктурі
Перевірив	Ролік О.І.			
Група	ІА-73мп			Діаграма послідовності взаємодії користувача з СУІ
Позначення документу	Додаток Ж			